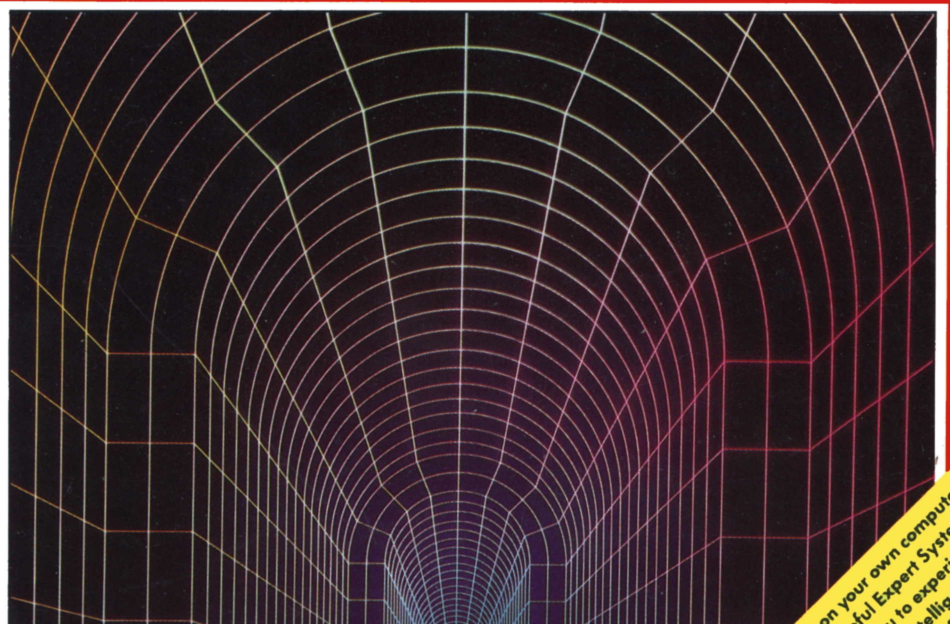
EXPLORING EXPERT SYSTEMS ON YOUR MICROCOMPUTER

Tim Hartnell



Explore the fascinating world of Expert Systems on your own computer. Contains a number of ready-to-run, genuinely useful Expert Systems programs, as well as BASIC interpreters to allow you to experiment with small-scale versions of the leading artificial intelligence languages — PROLOG and LISP. Machine-specific listings for SPECTRUM+ and SPECTRUM, AMSTRAD, BBC MICRO, COMMODORE 64, all MSX machines, and any machine furnished with Microsoft BASIC.

LONDON · MELBOURNE

EXPLORING EXPERT SYSTEMS ON YOUR MICROCOMPUTER

Listings provided for:

Spectrum/Spectrum+
Commodore 64
BBC Micro
Amstrad
All MSX machines

Any machine furnished with Microsoft BASIC

This work is dedicated to Bradley

EXPLORING EXPERT SYSTEMS ON YOUR MICROCOMPUTER

TIM HARTNELL

THE RIGHT LISTING FOR YOUR MACHINE:

The listings in the main body of the text are designed for the following computers:

AMSTRAD
APPLE
All MSX machines
Any computer with Microsoft BASIC

Towards the back of the book, you'll find machine-specific listings for:

COMMODORE 64
SPECTRUM + /SPECTRUM
BBC MICRO

INTERFACE PUBLICATIONS LTD., LONDON AND MELBOURNE

First published in the UK by:
Interface Publications Ltd,
9–11 Kensington High Street,
London W8 5NP, UK

First published in Australia by:
Interface Publications (Australia) Pty Ltd,
Chelsea House,
34 Camp Street,
Chelsea, Victoria, 3196

First printing August 1985

ISBN 0 907563 74 0

The programs in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. Whilst every care has been taken, the publishers cannot be held responsible for any running mistakes which may occur.

Books published by Interface Publications are distributed in the UK by WHS Distributors, St John's House, East Street, Leicester LE1 6NE (0533 551196) and in Australia and New Zealand by Pitman Publishing, Melbourne, Sydney, Brisbane and Wellington. Any queries regarding the contents of this volume should be directed by mail to Interface Publications at either the London or Chelsea, Australia, address.

Typeset by Wellset, Chelsea Heights, Melbourne, Australia
Cover design – Richard Kelly
Printed and Bound by Short Run Press Ltd, Exeter

Contents

Introduction

Welcome to this work on the fascinating world of expert systems. In it we'll be looking at specific milestones in the history and development of some major systems, with a view to getting an overall picture of the current expert systems' situation.

From there, we'll develop specific expert systems of our own (including THE AUTO MECHANIC, and MEDICI which will give you a quick 'stress-check' and predict how long you're likely to live). Our major expert system in this book delights in the name of FUZZY RITA. RITA gives you the framework of a general purpose expert system, which you can modify to serve you in any way you wish. RITA is shown in action sorting out cats from dogs (!), using the properties of a metal to tell you what it is, and finally, predicting the weather, using real data (and achieving a success rate far in excess of that which would happen by chance alone).

We'll also look at the languages which are beginning to dominate the Artificial Intelligence and expert systems' worlds. I've written BASIC emulators of LISP and PROLOG for you, which you can enter into your computer, to get a feel for working in these languages. A new and simple language, HASTE (for HArtnell's Simple declarative TonguE!) is given first, to lead you gently into the programming environments inhabited by LISP and PROLOG.

All in all, the book is designed to help you end up with a reasonably substantial 'knowledge base' of your own, so you can exercise your own expertise on expert systems.

Tim Hartnell,
London,
1985

Chapter One
What is an Expert System

An expert system is a computer program which contains human expertise on a subject, held in such a way that non-experts can have access to, and make use of, the expertise. Such a system is made up of two major parts — the stored information (the expertise) and the reasoning apparatus (which asks the user questions, and from the information it has been given, makes decisions).

Expert systems can do a lot. They can diagnose infectious diseases (MYCIN), deduce molecular structures from mass spectrograms (DENDRAL), look for oil and precious metals (PROSPECTOR) and help you start your car in the morning (THE AUTO MECHANIC, given later in this book).

The distinctive features of an expert/intelligent knowledge base system are these (and the terms should be easy to understand once you've worked through this book):

1 — Domain specific

2 — Uses 'fuzzy' or probabilistic reasoning

3 — Has a self-explanatory mode

4 — Clearly separates the facts from the inference mechanism

5 — Can grow incrementally

6 — Makes money

7 — Is typically rule-based

8 — Tends not to be over-ambitious

9 — Output is *advice* (such as 'change the spark plugs', 'operate on the left big toe', 'dig in the north-west corner of the orchard') rather than being the output of more data (such as a graph)

To tell the truth, we really don't know yet what an expert system really is. It's too early in the day to give a definitive answer to that apparently simple question: What is an expert system? We are unable to say which features, from the list we've just given, will turn out to be important, and which will end up being seen as peripheral.

Up to now, the biggest problem has been getting the knowledge from an expert into a form which a machine can manipulate. The current method, called 'knowledge engineering', is far from efficient, as it needs one human expert (the knowledge engineer) to laboriously extract the real knowledge which another human expert brings to bear on a problem. This is the real bottle-neck in the production of expert systems.

The Main Ingredients

Diagrammatically, here are the major parts of an expert system:

```
                                               Expert System
 Human: facts about the ]
        current problem ]---------> Listens to human
                                                 |
                                                \|/
                                             [ Inference engine
                                             [      /|\
 Human <--------------- Answer <-------------[       |
                                             [      \|/
                                             [ Knowledge base
```

The 'knowledge base', in a rule-based system (such as THE AUTO MECHANIC in this book), is basically a collection of IF/THEN statements such as:

IF X AND Y ARE TRUE THEN C IS TRUE

AUTO MECHANIC contains lines of coding which mean IF THE STARTER DOESN'T CLICK THEN THE SOLENOID IS PROBABLY FAULTY. It works by first asking the user to define the basic problem. AUTO MECHANIC then leads the user down a tree of possibilities, branching off from time to time as a result of the user input. This is an example of 'forward chaining', seeking out the unique solution which lies at the end of one particular series of branching.

Other, more sophisticated expert systems, use backward chaining, in which a goal is chosen (THE CAR HAS NO FUEL) and questions are asked to establish the truth of this particular hypothesus.

The Knowledge Base

This can be either encoded ('hard-wired') directly into the main body of the program (such as in MEDICI and AUTO MECHANIC) in which case it is more or less impossible to delete, or held within an addressable database, so that it can be accessed, modified and updated, even while the program is running. FUZZY RITA builds up a set of rules which determine the importance various inputs will be given, by assigning values to variables (and quite happily modifying these in the light of subsequent trials and feedback).

In the case of a hard-wired expert (and in stark contrast to the FUZZY RITA kind of program which generates its own rules while it is running, even if the user doesn't have a clue what the rules should be), the expertise must be acquired directly from a human expert.

A method by which a computer could extract the knowledge itself, and thus create its own knowledge base, will be a (perhaps, *the*) most important development in the history of expert systems. Self-created knowledge of this type will be quickly recognized as a very precious resource and commodity. Those who satisfy the demand for such a commodity will reap a fortune. (Capturing expertise from human experts is discussed in some detail in chapter seven, *Knowledge Engineering* of the book *Rule-Based Expert Systems* (Buchanan B., 1984).)

Several research projects are now underway around the world in the field of 'machine learning', the science of getting a computer to discover and encode the human expertise for itself. (For a fascinating account of how one knowledge engineer, H Penny Nii, works, see Feigenbaum, 1983.)

Here are some of the reasoning structures in use in present-day expert systems:

1 — DECISION TREES. The computer follows a hard path down a series of questions, letting the answer to each determine which specific channel the program should pursue next. This sort of a program needs *hard* data. It likes YES/NO situations. It is not able to cope easily with 'fuzzy' situations of the 'usually/perhaps/sometimes' type.

```
            o
        /       \
     o             o
   /   \         /   \
 o       o     o       o
```

Commercially-available programs such as CONSULT and EXPERT-EASE use decision trees.

2 — PRODUCTION RULES. This structure is seen in commercial programs such as XCON and DART, which use IF/THEN (along with AND) processes.

3 — SEMANTIC NETS. MYCIN and PROSPECTOR are examples of this type of structure:

```
          /-------- fact-one
         /          /        \
relationship-3     /      relationship-4
       /          /             \
fact-four        /            fact-three
                /                /
     relationship-1       relationship-2
                \               /
                 fact-three
```

Expert systems and artificial intelligence (A1) are hot topics at the moment. In the January 1985 edition of *Byte* magazine (pp. 275 — 282), under the heading *Expert Systems — Myth or Reality?,* Bruce D'Ambrosio points out that the US Department of Defense has singled out AI as one of the 10 most important technologies which should be developed in the closing years of this century. 'Japan Inc.' has swung into action with its 'fifth-generation computer project'. A major goal of this project is the development of better expert systems.

The major themes in research and development for fifth-generation computers (according to the *Proceedings of the International Conference on Fifth-Generation Computer Systems,* 19-22 October, 1981, Japan Information Processing and Development Centre) include the following:

BASIC APPLICATION SYSTEMS	Machine-translation Question-answering Applied speech-understanding Applied picture- and image-understanding Applied problem-solving
BASIC SOFTWARE SYSTEMS	Knowledge-base management Problem-solving and inference Intelligent interface
NEW ADVANCED ARCHITECTURE	Logic programming machine Functional machine Relational algebra machine Abstract data type support machine Data flow machine

DISTRIBUTED FUNCTION/NETWORK ARCHITECTURE	Database machine High-speed numerical computation machine High-level man-machine communication system
VLSI TECHNOLOGY	VLSI (very large-scale integrated) architecture Intelligent VLSI CAD (computer-aided design) architecture
SYSTEMISATION TECHNOLOGY	Intelligent programming system Knowledge-base design system

Spurred on by the Japanese efforts, France and Britain now have national projects underway. (Perhaps you can contribute to your own country's research with the expertise you gain from this book!) In the next chapter, we'll see how some of the ideas we've outlined are being used in practice.

Chapter Two
Expert Systems that Work

The best known, proven expert systems currently in use include the following three:

— MYCIN, which works in the field of infectious diseases

— DENDRAL, which analyzes organic chemical compounds; and

— PROSPECTOR, which suggests promising sites for mineral exploration

In this chapter, we'll look at all three. There are general lessons to be learned from them. As MYCIN appears to be the most important of them all, we will look at it more closely than we do the others.

The MYCIN Experiments at Stanford

We start with the Stanford expert system MYCIN, a powerful beast which can aid in the diagnosis of infectious diseases. MYCIN is one of the more successful expert systems in use at present, and there are many lessons which can be learned from studying it. From the information in this section, you may well see ways in which your own effective expert systems can be developed and expanded.

MYCIN is one of the oldest real expert systems in operation. It works on a rule-based method, in an area where it was not previously believed the data involved in decision-making could be codified.

MYCIN was preceded at the Stanford Heuristic Programming Project by DENDRAL, which can deduce molecular structures from analysing the output of a mass spectrograph. However, DENDRAL was not designed from the beginning to be an expert system; it evolved into one from an investigation into scientific induction. As well, DENDRAL works in a field where the processes involved in reaching a decision were fairly well-known before the system was built. MYCIN began as a project to build an expert system, in a field where the decision-making rules were fuzzy and ill-defined, and the project succeeded. (We will be looking at DENDRAL in some detail in due course.)

MYCIN is not designed to replace a doctor, but instead to act as a consultant. The patient is examined by a physician, who tells MYCIN the symptoms which have been observed. MYCIN then gives diagnostic advice, advice which is very similar to that which a specialist would give in the same circumstances. As well as dispensing advice, MYCIN can explain how it reached its conclusion. A development of this aspect of MYCIN's behaviour is an expert system which instructs medical students, thus creating new human experts from the encoded expertise from 'old' ones. (A prime source of information on the MYCIN project is *Rule-Based Expert Systems* (Buchanan, 1984). This is the book you should read if you wish to study the evolution of the project, and the thinking behind it, in detail.)

A further development of MYCIN, EMYCIN (for Essential MYCIN), was created by taking away the encoded diagnostic expertise (the 'knowledge base') and leaving the reasoning apparatus (the 'inference engine'). This has allowed expert systems on other subjects to be created, more or less simply by tacking new knowledge bases onto EMYCIN. This suggests that there are lessons in the structure of EMYCIN which can be generally applied to rule-based expert systems.

The MYCIN Gang

The developers of MYCIN (who eventually came to call themselves 'the MYCIN gang') started work on the system with the idea of following the lead of a program called SCHOLAR which was able to 'talk' very well on the geography of South America. However, they found that the medical data they were using did not fit as obviously into a neat framework as did the geographical facts, so the SCHOLAR line was abandoned.

Experimentation showed that the most fruitful developments lay in the direction of explicitly encoding specific rules — as used by human experts — which linked items in the database, rather than in trying to break these rules down to create general pathways which the decision-making process could follow. Holding the rules in this way also meant MYCIN could explain why it had come to the conclusion it had, using the kind of statements which physicians themselves would use. (Those involved with the development of MYCIN included J Aikins, S Axline, J Bennett, A Bonnet, B Buchanan, W Clancey, S Cohen, R Davis, L Fagan, F Rhame, C Scott, E Shortliffe, W vanMelle, S Wraith and V Yu.)

The Rules

The first real problem the gang had to come to grips with, once the nature of the rule base had evolved, was that human knowledge of infectious diseases did not fall into neat, mutually-exclusive packages. The barrier between one item and the next in the domain was often fuzzily defined.

The program's first implementation had 200 rules, which performed essentially as if they were lines being held in the form of an IF statement followed by a THEN (and sometimes including an ELSE). In later versions, the ELSEs were removed, as they were found to be rarely triggered. MYCIN now has more than 600 rules.

It was decided that MYCIN should make decisions in the following order:

- decide which organism (if any) was causing significant infection
- work out which organism it probably was
- decide which drugs could be of use
- suggest the best treatment

Here is a typical MYCIN rule:

```
If:(1)the stain of the organism is gram-positive, and
    (2)the morphology of the organism is coccus, and
    (3)the growth formation of the organism is clumps,
then there is suggestive evidence (0.7) that
   the identity of the organism is staphylococcus.
```

Note the number after the words 'suggestive evidence' (0.7) which indicates the degree of certainty which MYCIN is placing on the conclusion.

This degree of certainty is expressed on a scale which runs from -1 to +1, where -1 is the 'false' case (absolute disproof of an hypothesis), through 0 (there is no evidence either for or against the hypothesis, or the evidence points equally strongly both ways) to 1 (the hypothesis has been proven). The developers say this is an imperfect system, but is one which has worked significantly well in the past. They caution that it should be only be viewed as a starting point on the road towards the construction of a logical and consistent means of making decisions in areas where the data is uncertain, and the domain is complex (Shortliffe, 1983).

The rule is printed out for the physician in the easy-to-understand form given above. However, the rules are held internally as LISP lists, in the following way:

```
PREMISE:($AND (SAME CNTXT GRAM GRAMPOS)
              (SAME CNTXT MORPH COCCUS)
              (SAME CNTXT CONFORM CLUMPS))
ACTION: (CONCLUDE CNTXT IDENT STAPHYLOCOCCUS TALLY .7)
```

(The list above will probably make more sense to you once you've worked through the section in this book on EASLE and LISP-A.)

An expert system will often be used by people who are less expert than those whose expertise is encoded in the system. This means the reasoning behind the conclusions reached by the system will not always be apparent. MYCIN has a subprogram called CHRONICLER which can answer questions regarding the therapy it has advocated. A dialogue with CHRONICLER looks like this:

**** WHY DID YOU GIVE DRUG X FOR SITUATION Y?**
X was prescribed for Y
Since
— X is often used for Y in disease Z
— item Q of patient data shows intolerance to drug P
— X is not contraindicated

Knowledge engineering is a slow and expensive process. To allow MYCIN to improve its database by gaining new knowledge and weeding out errors, inconsistencies and omissions, a sub-program was developed called TEIRESIAS. When a physician comes across an error, or an incomplete diagnosis, TEIRESIAS can be triggered to ask a series of questions from which a new MYCIN rule can be created.

MYCIN has been designed to communicate with the consulting physician in as natural a form as possible, and this is clearly seen in a conversation with TEIRESIAS. Computer output during rule formation includes such 'human' lines as:

```
I hate to criticize, Dr ...., but did you know that
most rules in this area include ....

Shall I try to write a clause which covers...
```

Once the new rule has been added to the knowledge base, it is checked, with the physician being led gently through the checking process:

```
Rule xyz has now been added to the knowledge base.
I will now rerun the consultation to check the
effectiveness of your new rule. Get comfortable,
since this may take a bit.
```

Once the rules are in place, and have been tested and debugged, MYCIN compiles them into a tree structure, and then compiles this into machine code.

EMYCIN

MYCIN has two main components, the knowledge base and the inference engine. By 1974, two years after the MYCIN project began, the first experiments in replacing the diagnostic knowledge base with one in another field were underway. A *Pontiac Service Manual* was used as the source of expertise to give the inference engine material to work with. Fifteen rules relating to the horn circuit were written. (Although their structure is quite different, the nature of the knowledge locked into the AUTO MECHANIC program in this book closely resembles the rules given to this first 'son of MYCIN'.)

Experience in the *Pontiac* field, and further experiments, led to EMYCIN (Essential MYCIN) being developed. This is a MYCIN inference engine 'shell', along with a significant portion of the TEIRESIAS system (which, you'll recall, is used to add new rules to the knowledge base via a direct 'conversation' with a non-programmer user).

If you run EMYCIN with the intention of creating your own expert system, you are led through the creation process step by step, in a dialogue something like this:

```
> Do you wish to create a new knowledge base?
> Enter a word or phrase to describe your domain:
> Enter a one-word name for the root of your context
  tree, the central 'object' with which the consultation
  is concerned:
```

Once the information is in place, EMYCIN becomes an expert on that domain, and can be used as an expert system. So the program really has two, quite separate roles. The first role is that of accepting data and creating rules from it, and the second is one of making deductions from those rules.

Here are EMYCIN's roles in diagrammatic form:

```
                                                   ISSUE FOR
  +--HUMAN EXPERTISE---->--+                   +--E. TO SOLVE---+
 /|\                      \|/                 \|/              /|\
HUMAN                      |      *EMYCIN*     |              CLIENT
EXPERT               (CONSTRUCTION)  (CONSULTATION)              |
 /|\                       | /|\        /|\  \|/               /|\
  +--ARE MY RULES OK?--<---+  |          |    +------ADVICE-------+
                             \|/        \|/
                            +-+-DOMAIN--+-+
                            |  KNOWLEDGE  |
                            |    BASE     |
                            +-------------+
```

One of the major problems in creating new expert systems from EMYCIN is the lack of precision in many statements made in English. Of course, this lack of precision can be quantified to some extent in rule-making (as we saw with MYCIN), but it is not always possible to this, especially when EMYCIN may be interpreting the entered raw rule data incorrectly. An expert system (called ROGET) which assists in the creation of new expert systems with EMYCIN is now being developed by James Bennett (Buchanan, 1984) which should lead to more effective output from the sons of MYCIN.

Some very effective expert systems have already been developed from EMYCIN. These include SACON, CLOT and PUFF (Buchanan, 1984). SACON aims to uncover an *analysis strategy* for a particular problem involving a specific structure (in which the system infers such material deflection behaviors as excessive deflection and load path bifurcation!), while CLOT gives medical advice on bleeding problems. Another medical program, PUFF, generates advice on lung diseases, while two offspring of PUFF, CENTAUR and WHEEZE use PUFF's knowledge base, but manipulate this base with different representation and control structures from it. Another program, VM (for Ventilation Manager) keeps an eye on the post-operative condition of a patient after major cardiovascular surgery, when mechanical assistance with breathing is often required.

DENDRAL, A Chemical Whiz Kid

DENDRAL, which was written before MYCIN, is a program which analyzes organic compounds to determine their structure. The expert system, developed by E A Feigenbaum and J Lederberg at Stanford, is so effective at its task that it is believed that no human chemist could perform as well. Some of DENDRAL's analyses have even been published as original research results (Rich, 1983). It has also found errors in published chemical tables (Raphaele, 1976).

The program tries to infer the chemical composition and organic structure of chemical compounds, using data from a mass spectrogram of the substance, and nuclear magnetic resonance readings.

DENDRAL basically follows a three-step process on the way to determining the most probable structure for the compound it is analyzing. The three steps are plan, generate and test (Buchanan, 1981).

In the planning stage, the system uses its inbuilt rules to limit the kinds of structures which will be tested, eliminating ones which are incompatible with the data given. In the second stage, possible answers are produced, generated in accordance with DENDRAL's rule base. The final step consists of testing this structure. Without the planning stage, millions of possible structures could be generated, leading to the system taking an impracticable time to give an answer. In this stage, the possibilities are searched very meticulously, to ensure that there are solid chemical reasons for any discarding which takes place.

DENDRAL can locate the most important elements of the test data, and use these in conjunction with its basic knowledge of how chemical structures are formed. It tests its findings by an internal 'mass spectrometer simulator', which checks to see if the potential answer would produce the same spectrogram as the subtance being tested. Like MYCIN, DENDRAL uses a set of empirical rules given to it by human experts.

The mass spectrometer readings form one of the most important items of data with which DENDRAL works. A mass spectrometer separates atoms and molecules according to their mass, charging them electrically so that magnetic and/or electrical fields can be used to effect the separation. The spectrometer produces a graph, in which the relative heights of the peaks are related to the comparative populations of particular ions in the sample. (An easy-to-understand outline of the mass spectrometer is given in

Spectroscopy in Chemistry, Whitfield R C; London and Harlow, UK: Longmans, Green & Co., 1969; pp. 61 — 71.)

DENDRAL analyzes the spectrogram of a substance to be identified, and uses its condition-action rules to produce a list of substructures which must be in the unknown substance and a list that must not be.

Here is a DENDRAL rule (number 75; Winston, 1984). You can see it is of the same 'family' as the MYCIN rules we examined before:

If	**there is a high peak at 71 atomic mass units**
	there is a high peak at 43 atomic mass units
	there is a high peak at 86 atomic mass units
	there is any peak at 58 atomic mass units
then	**there must be an N-PROPYL-KETONE3 substructure**

DENDRAL'S generating mechanism is called CONGEN, and it takes input in the form of the readings from the mass spectrometer and other chemical tests. As well, the generator allows limits to be imposed on the structures it will produce, such as the number of atoms of each type in the molecule and any other constraints determined by the operator (Buchanan, 1981). From them, CONGEN outputs a complete list of all structures which make sense in light of all the entered data.

Although DENDRAL cannot be operated meaningfullly by those not highly trained in the particular areas of chemistry which constitute its domain, and thus could be said to have failed one text of an expert system ('does it make expertise available to non-experts?'), it performs better than the best human experts in the field, although it knows far less than the human experts (Simons, 1984). It has given worthwhile results in an extremely large and wide range of cases, including (Buchanan, 1981) determining the chemical structures of unknown compounds in the following areas:

— organic acids in human fluids
— impurities in manufactured chemicals
— antibiotics
— insect hormones and pheremones

Gold in Dem Dere Systems

PROSPECTOR sounds like a dream come true. Many reports of its actions suggest you simply describe an area to it, and it tells you where to dig for oil, gold or diamonds. But, of course, the reality (although exciting in its own way) is not quite as magical as some euphoric press accounts have suggested.

Despite this, some of PROSPECTOR's work has been very worthwhile. For example, the expert system was of major assistance in uncovering a potentially very rich offshoot from an existing molybdenum deposit near Mount Tolman in Washington (Winston, 1984). In this case, human experts disagreed with PROSPECTOR's findings, and only drilling proved that the system was right.

Over 500 rules and more than 300 assertions make up PROSPECTOR's rule-base. Its inference rules work along the same lines as MYCIN's production rules (Simons, 1984). However, while MYCIN can combine several possibilities (a patient may have more than one infection at once), PROSPECTOR uses strict probablisic inferences (see appendix A in this book on Bayes' Theorem), on the assumption that the system's findings are disjoint, that is, two outcomes cannot occur at once (Rich, 1983).

The knowledge held by PROSPECTOR was obtained by the usual laborious way of asking human experts (economic geologists, in this case) how they worked, and encoding this expertise. It is very interesting that every single one of the human experts involved said the knowledge engineering process enabled them to improve their own skills in this field, as they were forced to work out exactly what they did, and how they did it (Duda, 1981).

Chapter Three
Creating Expert Systems

There are several steps which are useful ground work for the development of a specific expert system. We start with the fairly obvious one of stating the problem. This can be done in exact terms, or in natural language ones. From this point, we need to describe the nature of the data inputs which will be used to reach a conclusion (the output of the expert system).

Using this information, it's possible to work out the theoretical mechanisms by which the problem will be solved (the 'symbolic calculi' which will be brought to bear).

Now we get to the hard part, actually turning out perception of the problem, and our desire for a particular reasoning path upon which a solution will be found, into a procedure (or series of them), which will work in practice. It is useful, at this stage, to consider the incorporation of a 'reporting mechanism' in the system, which can tell a human observer why certain paths are being followed and — at the end — explain how the system reached its final conclusion. The procedures, ideally, will have self-modification elements, which ensure that fruitless paths are quickly terminated, thus saving processing strength and time for avenues which are more likely to have some real bearing on the problem.

The two fundamental, and reasonably obvious, conditions which must be fulfilled if any expert system you develop is to be really useful are as follows. The first one is that the system must do what it claims. If you have built an expert system to distinguish between various types of sea-shells, it must be able to exhibit adequate reasoning powers when confronted with the vast majority of objects within its domain. Secondly, no matter how 'clever' the system is in this regard, it is unlikely to be used if it is not written to make interaction with the user reasonably simple.

While it is fairly easy to get to grips with, say, a personal accounts computer program, using an expert system demands more than just a knowledge of the format of data inputs, and the expected outputs. An expert system is a complex, but limited, device. To use it properly, the user needs to understand the limits within which the program can perform.

It's likely that a user will need to spend a lot of time with the program before he or she really begins to understand its possibilities and restrictions. The on-screen presentation, and the program documentation, should do all they can to limit the amount of time which must be spent before the program can be used in a meaningful way.

Information Types

Of course, the general domain of the program will be known ('vitamin deficiencies') and the context of inputs ('rough skin', 'tendency to bruise easily') and outputs ('the evidence indicates a marked lack of Vitamin A in the diet') will be fairly easily appreciated with most systems. However, the kind of information the program is able to cope with may take some time to determine.

Sometimes, when developing an expert system, it is very useful to be able to tell the system that some answers are more important than others. Imagine a medical diagnosis program, which was interested in back problems. An answer to a question such as "Does the patient suffer pain when trying a lift a weight?" could well be more important than the answer to a question such as "Does the patient eat green, leafy vegetables each day?". In such a case, the expert system needs a method which will enable it to give emphasis to important answers, and in other circumstances, to play down the answers to some questions.

For this, we use what are known as *logical sufficiency* values for answers which should be weighted, and *logical necessity* values for those answers which should be played down. To use them, you *multiply* an answer by the logical sufficiency value if it is *true,* and *divide* by the logical necessity number if it is *false.* This ensures the system will tend to take more notice of a single answer which matters than allowing it to be swamped, so to speak, with ten answers which do not matter much at all.

```
                    +------+
                    | goal |
                    +------+
                   /        \
                  /          \
+------------+         +------------+
|  partial   |         |  partial   |
| conclusion |         | conclusion |
+------------+         +------------+
 /          \           /           \
/            \      /LS 2           \LS 10
                   /LN 0.3           \LN 0.01
                  /                   \
          +-------------+     +-------------+
          |  question:  |     |  question:  |
          |  answer on  |     |  answer on  |
          | statistical |     | statistical |
          |    scale    |     |    scale    |
          +-------------+     +-------------+
```

Imagine we have developed a medical diagnosis expert system which follows a decision tree, with questions and rules at the leaf nodes. This kind of expert system can start from goal statements, ask appropriate questions, and then use its rule-base to make decisions.

Our system is set up to ask the relevant questions ('Is the patient sweating profusely'), accept the answer as a statement chosen from a menu of possibilities ("YES", through "I DON'T KNOW", to "NO"), convert this to a probability, then multiply the true answers by the logical sufficiency value for that question, and divide false answers by the logical necessity value for that question.

It is often desirable for an expert system to contain a mechanism which will stop certain lines of questioning, if specific answers given earlier indicate that some lines of questioning are meaningless. Following "IS THE ENGINE LYING IN BITS ALL OVER THE GROUND" (which gets, let us say, a "YES" answer) with a question like "ATTEMPT TO START THE CAR; DOES MOTOR TURN OVER?" wastes time, gets answers which cannot be assessed in working towards a conclusion, and demonstrates convincingly the lack of intelligence the system really possesses.

As well as a means of stopping certain questions being asked later in the run as a result of answers given earlier, we need to be able to ask some questions later, *only if* positive answers were received to other specific questions earlier.

Two Types of Reasoning

There appear to be two broad categories into which reasoning activities can be placed. Knowledge of these categories makes it simplier to create specific expert systems, so long as it is possible (and it nearly always is) to decide which category the particular class of problems — to be solved by our expert system — will fit into.

For want of better descriptions, I prefer to call the categories 'exact' and 'approximate' reasoning. A logical problem which is exact is one where there is — given all the inputs — one 'true' output. An approximate problem is one in which the inputs are weighted numerically as to their 'trueness', with the output also qualified numerically.

As you now know, the two major parts of many expert systems (such as the infectious diseases diagnostic program, MYCIN, which we looked at in detail in the previous chapter) are called the *knowledge base* (the stored expertise) and the *inference engine* (the reasoning mechanism). We have seen that a great deal of the information we know about the world is not capable of being reduced to numbers. Our lives are riddled with situations where phrases like *it seems likely, often, sometimes* and *I don't know why, but I get a feeling that you'll discover the problem if you* ... For expert systems to be of real use in the non-ideal, non-numerically-discrete world we inhabit, they must be capable of working with human concepts and perception.

Human beings do not, as a rule, reason only (or even principally) by following rules. In many cases, the weight given to certain elements within a logical proposition is based on 'real-world knowledge'.

This is very clearly demonstrated in the understanding of written or spoken text. The words in such a text have meaning only within a very wide real-world setting. The extent of our knowledge of the environment within which the text exists, and which it reflects and models, dictates the degree of understanding we can bring to the text. This statement alone suggests how difficult it is to develop expert systems which interact with us in natural language. It is, with current technology, impossible to instill more than a minute fraction of the richness of our own real-world knowledge into a machine. And without this richness, natural language understanding can only be effective (and not very effective at that) within very narrow domains.

Chapter Four
Rule-based Systems

A primitive rule-based expert system can be created fairly easily. Such a system, perhaps the simplest kind of expert system you can write, can nevertheless be effective and useful, as well as giving an insight into how other, more complex rule-based systems such as the infectious disease diagnostic program MYCIN, have been created.

In this chapter, I'll introduce you to THE AUTO MECHANIC, a simple rule-based system which is designed to help hopeless non-mechanics (such as myself) focus on possible causes of problems they are having with their cars. Because this program is based on real information painlessly extracted from human experts in this field, you may well find this program can be of genuine use to you.

However, the main reason for including it in the book is to show how a rule-based system can be developed, in which rules are encoded in a specific order, and the answer to one question dictates which question the system will ask next.

Sides to Every Rule

The simplest rule for use in an expert system has two parts. They are a left hand side (LHS) which is typically an IF statement (IF THE COMPUTER IS ON FIRE), and a right had side (RHS) which is usually a THEN clause (THEN SQUIRT IT WITH THE EXTINGUISHER). A low-level rule-based expert system could be little more than a series of LHS-RHS pairs which are presented in a specific order.

A primitive system of this type would allow the user to enter the answer to a question (such as IS THE COMPUTER ON FIRE, SITTING THERE SERENELY OR MELTING INTO AN UGLY BLOB) and then search through its collection of rules for a match between the answer given and the LHS of a rule. At this point it would either output the relevant RHS, or use the RHS to direct to a further, more specific question.

The order in which the collection of rules is stored is important, as is the way in which LHS matchings are handled. If there are several LHS conditions which must be met before a RHS is envoked, the system must know such things as how many matches are required and how close an answer must be to constitute a match. In some expert systems, it will make sense for a RHS to be envoked the moment a match is found. In other systems, where a single answer from the user may match several LHS's, these could be arranged in heirachical order, with the higher level ones envoked if possible before the lower level ones were even investigated. DENDRAL, a rule-based system which produces information for chemists from mass spectrometry data, works with an heirachy of rules.

In THE AUTO MECHANIC, the rules are in a strict order, and there is a single, clear path to the next rule. This path is effectively the RHS of the rule.

Let's see it in action. It starts with a menu of possibilities:

```
SO YOU'RE HAVING AUTOMOBILE PROBLEMS.
LET'S SEE IF WE CAN PIN DOWN
THE TROUBLE...
```

```
ENTER A LETTER WHICH DESCRIBES PROBLEM:

   A - ENGINE WON'T TURN OVER
   B - ENGINE TURNS, BUT WON'T START
   C - STARTS, THEN STALLS STRAIGHT AWAY
   D - CAR RUNS, BUT STALLS A LOT
   E - CAR RUNS, BUT IDLES ROUGHLY
```

Let's say our problem is the second one: ENGINE TURNS, BUT WON'T START. Once we've told it this, it leads us a through number of checkpoints, and emergency repair suggestions:

```
                              > OK    B <

   CHECK WIRES AT POINTS FOR DAMPNESS.
   IS THERE A CHANCE AREAS ARE DAMP?
                    (Y - YES, N - NO)?
                              > OK    Y <

   SPRAY WITH MOISTURE REMOVAL SPRAY

   IS THERE DUST VISIBLE ON THE INSULATING
        PART OF THE COIL, OR ON THE
             DISTRIBUTOR CAP?
                    (Y - YES, N - NO)?
                              > OK    Y <

   WIPE COIL SECTION AS WELL AS INSIDE
        AND OUTSIDE OF CAP.

   ENSURE ALL WIRES ARE TIGHT AND DRY
             BEFORE CONTINUING
```

```
IF CAR STILL DOES NOT START, IT IS TIME
TO CHECK THE SPARK PLUGS:

DOES SPARK FROM THE END OF THE PLUG WIRE
JUMP 3/8 INCH OR MORE?
                    (Y - YES, N - NO)?
                         > OK    N <
```

The order in which the expert system gives the suggestions, and which suggestions it makes, depends upon the answers you give:

```
IS THERE ANY SPARK AT ALL?
                    (Y - YES, N - NO)?
                         > OK    Y <

ARE PLUGS GREASY?
                    (Y - YES, N - NO)?
                         > OK    N <

IF THIS IS AN EMERGENCY, TRY CLOSING
THE GAP TO ABOUT HALF NORMAL

DO YOU HAVE GAS IN THE TANK?
                    (Y - YES, N - NO)?
                         > OK    Y <

ARE ALL FUEL AND VACUUM LINES SECURE?
                    (Y - YES, N - NO)?
                         > OK    Y <

   REMOVE THE AIR CLEANER FROM THE
            CARBURETOR
```

```
DOES IT LOOK DRY?
                    (Y - YES, N - NO)?
                         > OK   N <

HAVE YOU BEEN CRANKING THE STARTER A
LOT IN THE PAST FEW MINUTES?
                    (Y - YES, N - NO)?
                         > OK   Y <

WAIT FOR A MINUTE OR SO, THEN HOLD THE
GAS PEDAL STEADILY ON THE FLOOR WITHOUT
PUMPING IT. THIS SHOULD GET YOU GOING
```

If you ever have trouble starting your car, and you'd like your computer to save you from having to call an emergency repair service, enter the following listing, and you'll have an AUTO MECHANIC at your beck and call:

```
10 REM THE AUTO MECHANIC
20 CLS
30 PRINT "SO YOU'RE HAVING AUTOMOBILE PR
OBLEMS."
40 PRINT "LET'S SEE IF WE CAN PIN DOWN
          THE TROUBLE..."
50 PRINT
60 PRINT "ENTER A LETTER WHICH DESCRIBES
 PROBLEM:"
70 PRINT
80 PRINT "  A - ENGINE WON'T TURN OVER"
90 PRINT "  B - ENGINE TURNS, BUT WON'T
START"
100 PRINT "  C - STARTS, THEN STALLS STR
AIGHT AWAY"
110 PRINT "  D - CAR RUNS, BUT STALLS A
LOT"
```

```
120 PRINT "  E - CAR RUNS, BUT IDLES ROU
GHLY"
130 IF INKEY$<>"" THEN 130
140 A$=INKEY$
150 IF A$<"A" OR A$>"E" THEN 140
160 GOSUB 1560
170 ON ASC(A$)-64 GOTO 200,580,1200,1230
,1350
180 END
190 REM **************
200 REM WON'T TURN OVER
210 PRINT "WE'LL START BY CHECKING THE B
ATTERY."
220 PRINT "TURN ON THE LIGHTS."
230 PRINT TAB(6);"ARE THEY DIM?"
240 GOSUB 1520
250 IF A$="N" THEN 350:REM BATTERY OK
260 PRINT "ARE BATTERY CABLES LOOSE OR C
ORRODED?"
270 GOSUB 1520
280 IF A$="Y" THEN PRINT "TIGHTEN AND CL
EAN"
290 GOSUB 1580
300 PRINT "IS FAN BELT LOOSE?"
310 GOSUB 1520
320 IF A$="Y" THEN PRINT TAB(8);"TIGHTEN
 THE FAN BELT"
330 GOSUB 1580
340 PRINT "JUMPER LEADS OR A PUSH SHOULD
 BE ENOUGH TO START THE CAR":END
350 PRINT "IS THERE LOOSENESS OR CORROSI
ON AT THE"
360 PRINT "STARTER END OF THE BATTERY CA
BLE"
```

```
370 GOSUB 1520
380 IF A$="Y" THEN PRINT "TIGHTEN AND CL
EAN THE CONNECTIONS"
390 GOSUB 1580
400 PRINT "PUT A PIECE OF METAL ACROSS T
HE"
410 PRINT "SOLENOID TERMINALS. DOES STAR
TER WORK?"
420 GOSUB 1520
430 IF A$="Y" THEN PRINT "THE IGNITION S
WITCH IS PROBABLY FAULTY"
440 GOSUB 1580
450 IF A$="Y" THEN PRINT TAB(4);"IT SHOU
LD BE REPLACED":END
460 PRINT "DOES STARTER CLICK?"
470 GOSUB 1520
480 IF A$="Y" THEN PRINT "THE STARTER MA
Y BE JAMMED":GOTO 520
490 PRINT "AS NOTHING HAPPENED, THE SOLE
NOID IS"
500 PRINT "PROBABLY FAULTY. A PUSH MAY S
TART"
510 PRINT TAB(12);"YOUR CAR.":END
520 REM STARTER JAMMED
530 PRINT "TURN THE IGNITION OFF, AND PU
T THE CAR"
540 PRINT "IN A HIGH GEAR. PUSH CAR FOR
A FOOT OR"
550 PRINT "SO TO POP STARTER LOOSE.":END

560 RETURN
570 REM ***************
580 REM TURN OVER, WON'T START
590 PRINT "CHECK WIRES AT POINTS FOR DAM
PNESS."
```

```
600 PRINT "IS THERE A CHANCE AREAS ARE D
AMP?"
610 GOSUB 1520
620 IF A$="Y" THEN PRINT "SPRAY WITH MOI
STURE REMOVAL SPRAY":GOSUB 1580
630 PRINT "IS THERE DUST VISIBLE ON THE
INSULATING"
640 PRINT TAB(6);"PART OF THE COIL, OR O
N THE"
650 PRINT TAB(11);"DISTRIBUTOR CAP?"
660 GOSUB 1520
670 IF A$="Y" THEN PRINT "WIPE COIL SECT
ION AS WELL AS INSIDE"
680 IF A$="Y" THEN PRINT TAB(6);"AND OUT
SIDE OF CAP.":GOSUB 1580
690 PRINT "ENSURE ALL WIRES ARE TIGHT AN
D DRY                  BEFORE CONTINUING"
700 GOSUB 1580
710 PRINT "IF CAR STILL DOES NOT START,
IT IS TIME TO CHECK THE SPARK PLUGS:"
720 GOSUB 1580
730 PRINT "DOES SPARK FROM THE END OF TH
E PLUG WIREJUMP 3/8 INCH OR MORE?"
740 GOSUB 1520
750 IF A$="Y" THEN PRINT "THE PLUGS ARE
FAULTY":GOSUB 1580
760 IF A$="N" THEN 830
770 PRINT "ARE PLUGS GREASY?"
780 GOSUB 1520
790 IF A$="Y" THEN PRINT "AN EMERGENCY R
EPAIR CANNOT BE MADE"
800 IF A$="Y" THEN PRINT "WHILE PLUGS AR
E IN THAT STATE.":GOSUB 1580:GOTO 770
810 IF A$="N" THEN PRINT "IF THIS IS AN
EMERGENCY, TRY CLOSING"
```

```
820 IF A$="N" THEN PRINT "THE GAP TO ABO
UT HALF NORMAL":GOSUB 1580:GOTO 910
830 PRINT "IS THERE ANY SPARK AT ALL?"
840 GOSUB 1520
850 IF A$="Y" THEN 770
860 GOSUB 870:END
870 PRINT "CHECK ROTOR, COIL AND DISTRIB
UTOR CAP"
880 PRINT "FOR CRACKS. IF THERE AREN'T A
NY THERE,"
890 PRINT "IT LOOKS AS IF THE POINTS OR
CONDENSER  IS YOUR PROBLEM"
900 PRINT "A REPAIR MAY WELL BE NEEDED":
RETURN
910 PRINT "DO YOU HAVE GAS IN THE TANK?"

920 GOSUB 1520
930 IF A$="N" THEN PRINT "FILL TANK AND
TRY AGAIN":GOSUB 1580
940 PRINT "ARE ALL FUEL AND VACUUM LINES
 SECURE?"
950 GOSUB 1520
960 IF A$="N" THEN PRINT "ATTEND TO THES
E AND TRY AGAIN":GOSUB 1580
970 PRINT "    REMOVE THE AIR CLEANER FRO
M THE"
980 PRINT TAB(13);"CARBURETOR":GOSUB 158
0
990 PRINT "DOES IT LOOK DRY?"
1000 GOSUB 1520
1010 IF A$="N" THEN 1080
1020 PRINT "TURN THE ENGINE OVER A FEW T
IMES WITH"
1030 PRINT "YOUR HAND SEALING THE AIR IN
TAKE":GOSUB 1580
1040 PRINT "IS YOUR HAND WET WITH GAS?"
```

```
1050 GOSUB 1520
1060 IF A$="N" THEN PRINT "UNSCREW GAS C
AP IN CASE AIR VENT IS PLUGGED"
1070 IF A$="N" THEN PRINT "THE FUEL PUMP
 MAY NOT BE WORKING":GOSUB 900:END
1080 PRINT "HAVE YOU BEEN CRANKING THE S
TARTER A"
1090 PRINT "LOT IN THE PAST FEW MINUTES?
"
1100 GOSUB 1520
1110 IF A$="N" THEN 1150
1120 PRINT "WAIT FOR A MINUTE OR SO, THE
N HOLD THE "
1130 PRINT "GAS PEDAL STEADILY ON THE FL
OOR WITHOUT"
1140 PRINT "PUMPING IT. THIS SHOULD GET
YOU GOING":END
1150 PRINT "THE ENGINE MAY WELL BE FLOOD
ED AT THIS"
1160 PRINT "POINT AND THE FLOAT VALVE ST
UCK OPEN.":GOSUB 1580
1170 PRINT "TAP THE SIDE OF THE CARBURET
OR"
1180 PRINT "THEN TRY THE STARTING PROCES
S AGAIN":END
1190 REM **************
1200 REM STARTS, THEN STALLS
1210 PRINT "THIS SUGGESTS A FAULTY BALLA
ST RESISTOR WHICH SHOULD BE REPLACED":EN
D
1220 REM **************
1230 REM RUNS, STALLS A LOT
1240 PRINT "THE PROBLEM IS EITHER CAUSED
 BY:"
```

```
1250 PRINT TAB(7);"SHORTING (OR LOOSE) W
IRES;"
1260 PRINT TAB(7);"A WEAK SPARK; OR"
1270 PRINT TAB(7);"A FAULT IN THE FUEL S
YSTEM"
1280 PRINT "CHECK FIRST FOR LOOSE OR SHO
RTING WIRES":GOSUB 1580
1290 PRINT "IF THEY ARE NOT OK, REPAIR.
IF THEY ARE,IT COULD BE THE SPARK PLUGS"

1300 GOSUB 870
1310 PRINT "THERE IS A FINAL CHECK WE CA
N TRY ON"
1320 PRINT "YOUR SPARK PLUGS":GOSUB 1580

1330 GOTO 1360
1340 REM ***************
1350 REM RUNS, ROUGH IDLE
1360 PRINT "IT COULD WELL BE THAT ONE OR
 MORE OF"
1370 PRINT "YOUR SPARK PLUGS ARE FAULTY.
":GOSUB 1580
1380 PRINT "DISCONNECT THEM ONE AT A TIM
E. THE ONES"
1390 PRINT "WHICH DO NOT CAUSE THE ENGIN
E IDLE TO"
1400 PRINT "DROP ARE FAULTY. CHECK THESE
 THEN"
1410 PRINT "RETURN TO THE SYSTEM.":GOSUB
 1580
1420 PRINT "DID TEST SHOW ANY PLUGS WERE
 FAULTY?"
1430 GOSUB 1520
1440 IF A$="Y" THEN PRINT "REPLACE ALL P
LUGS IF YOU CAN, OR JUST"
```

```
1450 IF A$="Y" THEN PRINT "THE ONES WHIC
H TESTED FAULTY":GOSUB 1580
1460 PRINT "THE MOST COMMON CAUSE OF A B
AD IDLE,"
1470 PRINT "ASSUMING THAT THE PLUGS ARE
OK, IS THAT"
1480 PRINT "YOUR GAS MIXTURE IS SET TOO
RICH"
1490 PRINT "SO YOU SHOULD ADJUST THIS":E
ND
1500 PRINT "ADJUST THE IDLE SPEED-SCREW
ON THE       THROTTLE LINKAGE":END
1510 REM **************
1520 PRINT TAB(16);"(Y - YES, N - NO)?"
1530 IF INKEY$<>"" THEN 1530
1540 A$=INKEY$
1550 IF A$<>"Y" AND A$<>"N" THEN 1540
1560 PRINT TAB(22);"> OK   ";A$;" <"
1570 BEEP:REM ADD 'BEEP' OR SIMILAR
          SOUND IN THIS LINE
1580 FOR T=1 TO 1000:NEXT T
1590 PRINT
1600 RETURN
```

As well as seeing how a rule-based system can work by using this program, I hope you can appreciate one of the real disadvantages of having a system in which the rules are 'hard-wired'. They are extremely difficult to modify, without almost totally destroying the program. Tracking down an error of judgement by the system could also be difficult, and there is no possibility of the system 'learning on the job', as other programs (such as FUZZY RITA, which we come to later in this book) can do.

Despite these disadvantages, in many situations a purpose-built, hard-wired system like THE AUTO MECHANIC is far more useful within its domain than would be a general system which had simply been taught about the subject.

You need to weigh up these advantages and disadvantages when determining which kind of expert will best serve your specific needs.

Chaper Five
The Doctor Is In

In my book *Exploring Artificial Intelligence on your Microcomputer* (Interface Publications Ltd., UK, 1984) I introduced a fictitious expert system called MEDICI, which asked a number of questions and from the answers gave specific advice on health and longevity. I decided to actually create a MEDICI for this book.

The best way to live for a long time is to choose parents and grand-parents with long lives. As that removes the element of choice from the matter, it is up to most of us to accept the second-best way. This is to live from day-to-day applying the knowledge of nutrition and health which man has acquired. Of course, few of us live as intelligently as we could, or apply all the information we've come across. If nothing else, that would probably be a pretty boring way to live.

MEDICI will allow you to tell it how you really live, rather than ask you about the health information which you believe, but do not consistently apply to your life-style. And once the expert system has given you a health 'rating' and commented on your health status, it will indicate the range in which your life expectancy may well lie. You can then rerun the system, and see what changes you would have to make in your life-style in order to increase the number of years you have in which to write expert systems on your computer.

Let's run MEDICI past someone (an unwilling volunteer from my office, actually) and see what it comes up with:

```
WHICH OF THE FOLLOWING IS CLOSEST
TO THE TRUTH (SELECT ONE):

A - I AM BADLY OVERWEIGHT
B - I AM FAIRLY OVERWEIGHT
C - I AM SLIGHTLY OVERWEIGHT
D - MY WEIGHT IS ABOUT RIGHT
E - I AM THINNER THAN I SHOULD BE

            OK    C
-5


WHICH OF THE FOLLOWING IS CLOSEST
TO THE TRUTH (SELECT ONE):

I ENGAGE IN EXERCISE, THAT
RAISES MY HEARTBEAT TO 120 OR MORE,
FOR AT LEAST THE FOLLOWING NUMBER
     OF HOURS A WEEK:

A - LESS THAN A QUARTER
B - MORE THAN A QUARTER, UP TO
    THREE-QUARTERS
C - FROM THREE-QUARTERS OF AN HOUR
    UP TO ONE AND A HALF
D - FROM ONE AND A HALF TO
    TWO AND A HALF
E - MORE THAN TWO AND A HALF HOURS

            OK    A
 0
```

WHICH OF THE FOLLOWING IS CLOSEST
TO THE TRUTH (SELECT ONE):

WHEN DRIVING:

A - I HARDLY EVER WEAR A SEAT BELT
B - I WEAR A SEAT BELT AROUND A QUARTER
    OF THE TIME
C - I WEAR A SEAT BELT EVERY SECOND
    JOURNEY
D - I WEAR A SEAT BELT FOR MOST, BUT NOT
    ALL TRIPS
E - I ALWAYS WEAR A SEAT BELT

OK E

8

WHICH OF THE FOLLOWING IS CLOSEST
TO THE TRUTH (SELECT ONE):

I AM CONSCIOUS OF NUTRITION AND TRY
TO EAT HEALTHILY:

A - ALL THE TIME
B - NEARLY ALL THE TIME
C - A FAIR PROPORTION OF THE TIME
D - FROM TIME TO TIME
E - HARDLY AT ALL

OK D

1

```
WHICH OF THE FOLLOWING IS CLOSEST
TO THE TRUTH (SELECT ONE):

SMOKING (A CIGAR COUNTS AS A CIGARETTE)

A - NOT AT ALL
B - LESS THAN 15 CIGARETTES A DAY
C - 15 TO 25 CIGARETTES A DAY
D - 26 TO 42 CIGARETTES A DAY
E - MORE THAN 42 CIGARETTES A DAY

             OK    A
 0


WHICH OF THE FOLLOWING IS CLOSEST
TO THE TRUTH (SELECT ONE):

ALCOHOL - HOW MANY DRINKS (ON AVERAGE)
DO YOU HAVE EACH DAY?

A - NONE
B - LESS THAN 3
C - 3 TO 6
D - 7 TO 9
E - MORE THAN 9

             OK    C
-6
```

```
PERSONAL ASSESSMENT FROM MEDICI:

        WEIGHT:-5
      EXERCISE: 0
   CAR SAFETY: 8
    NUTRITION: 1
      SMOKING: 0
      ALCOHOL:-6
       STRESS:-8

    YOUR RAW RATING IS -10

  ON A SCALE WHERE ZERO IS AVERAGE,
THE LOWEST RATING IS BELOW -80, AND
   THE HIGHEST IS OVER 30

THIS INDICATES YOUR HEALTH STATUS
IS BELOW AVERAGE

LIFE EXPECTANCY:
  MALE         FEMALE
  60 TO 66     65 TO 71
```

Here's the listing, so you can check on your current health status:

```
10 REM MEDICI - PERSONAL CHECKUP
20 CLS
30 GOSUB 1250
40 PRINT "A - I AM BADLY OVERWEIGHT"
50 PRINT "B - I AM FAIRLY OVERWEIGHT"
60 PRINT "C - I AM SLIGHTLY OVERWEIGHT"
70 PRINT "D - MY WEIGHT IS ABOUT RIGHT"
80 PRINT "E - I AM THINNER THAN I SHOULD
 BE"
90 GOSUB 1170
```

```
100 WEIGHT=5*(ASC(A$)-68):IF A$="E" THEN
 WEIGHT=0
110 PRINT WEIGHT
120 GOSUB 1250
130 PRINT "I ENGAGE IN EXERCISE, THAT"
140 PRINT "RAISES MY HEARTBEAT TO 120 OR
 MORE,"
150 PRINT "FOR AT LEAST THE FOLLOWING NU
MBER"
160 PRINT TAB(8);"OF HOURS A WEEK:"
170 PRINT
180 PRINT "A - LESS THAN A QUARTER"
190 PRINT "B - MORE THAN A QUARTER, UP T
O              THREE-QUARTERS"
200 PRINT "C - FROM THREE-QUARTERS OF AN
 HOUR          UP TO ONE AND A HALF"
210 PRINT "D - FROM ONE AND A HALF TO
               TWO AND A HALF"
220 PRINT "E - MORE THAN TWO AND A HALF
HOURS"
230 GOSUB 1170
240 EXERCISE=5*(ASC(A$)-63)-5:IF A$="A"
THEN EXERCISE=0
250 PRINT EXERCISE
260 GOSUB 1250
270 PRINT "WHEN DRIVING:":PRINT
280 PRINT "A - I HARDLY EVER WEAR A SEAT
 BELT"
290 PRINT "B - I WEAR A SEAT BELT AROUND
 A QUARTER     OF THE TIME"
300 PRINT "C - I WEAR A SEAT BELT EVERY
SECOND         JOURNEY"
310 PRINT "D - I WEAR A SEAT BELT FOR MO
ST, BUT NOT    ALL TRIPS"
320 PRINT "E - I ALWAYS WEAR A SEAT BELT
"
```

```
330 GOSUB 1170
340 SEATBELT=2*(ASC(A$)-65)
350 PRINT SEATBELT
360 GOSUB 1250
370 PRINT "I AM CONSCIOUS OF NUTRITION A
ND TRY      TO EAT HEALTHILY:"
380 PRINT
390 PRINT "A - ALL THE TIME"
400 PRINT "B - NEARLY ALL THE TIME"
410 PRINT "C - A FAIR PROPORTION OF THE
TIME"
420 PRINT "D - FROM TIME TO TIME"
430 PRINT "E - HARDLY AT ALL"
440 GOSUB 1170
450 DIET=-ASC(A$)+69
460 PRINT DIET
470 GOSUB 1250
480 PRINT "SMOKING (A CIGAR COUNTS AS A
CIGARETTE)"
490 PRINT
500 PRINT "A - NOT AT ALL"
510 PRINT "B - LESS THAN 15 CIGARETTES A
 DAY"
520 PRINT "C - 15 TO 25 CIGARETTES A DAY
"
530 PRINT "D - 26 TO 42 CIGARETTES A DAY
"
540 PRINT "E - MORE THAN 42 CIGARETTES A
 DAY"
550 GOSUB 1170
560 SMOKING=-7*(ASC(A$)-65)
570 PRINT SMOKING
580 GOSUB 1250
590 PRINT "ALCOHOL - HOW MANY DRINKS (ON
 AVERAGE)  DO YOU HAVE EACH DAY?"
```

```
600 PRINT
610 PRINT "A - NONE"
620 PRINT "B - LESS THAN 3"
630 PRINT "C - 3 TO 6"
640 PRINT "D - 7 TO 9"
650 PRINT "E - MORE THAN 9"
660 GOSUB 1170
670 DRINK=-30
680 IF A$="A" THEN DRINK=0
690 IF A$="B" THEN DRINK=1
700 IF A$="C" THEN DRINK=DRINK/5
710 IF A$="D" THEN DRINK=DRINK/2
720 PRINT DRINK
730 GOSUB 1250
740 PRINT "IN GENERAL, HOW STRESSFUL WOU
LD YOU SAY";
750 PRINT "YOUR LIFE HAS BEEN IN THE PAS
T 6 MONTHS"
760 PRINT
770 PRINT "A - EXTREMELY STRESSFUL"
780 PRINT "B - FAIRLY STRESSFUL"
790 PRINT "C - SLIGHTLY STRESSFUL"
800 PRINT "D - NEUTRAL"
810 PRINT "E - NOT STRESSFUL"
820 GOSUB 1170
830 STRESS=INT(2.5*(ASC(A$)-69))
840 PRINT STRESS
850 GOSUB 1300:CLS
860 PRINT "PERSONAL ASSESSMENT FROM MEDI
CI:"
870 PRINT
880 PRINT TAB(8);"WEIGHT:"WEIGHT
890 PRINT TAB(6);"EXERCISE:"EXERCISE
900 PRINT TAB(4);"CAR SAFETY:"SEATBELT
910 PRINT TAB(5);"NUTRITION:"DIET
920 PRINT TAB(7);"SMOKING:"SMOKING
```

```
930 PRINT TAB(7);"ALCOHOL:"DRINK
940 PRINT TAB(8);"STRESS:"STRESS
950 GOSUB 1300
960 ANT=WEIGHT+EXERCISE+SEATBELT+DIET+SM
OKING+DRINK+STRESS
970 GOSUB 1300:PRINT
980 PRINT "     YOUR RAW RATING IS "ANT:P
RINT
990 PRINT "  ON A SCALE WHERE ZERO IS AV
ERAGE,"
1000 PRINT "THE LOWEST RATING IS BELOW -
80, AND"
1010 PRINT "    THE HIGHEST IS OVER 30"
1020 GOSUB 1300:PRINT
1030 IF ANT<6 AND ANT>-6 THEN A$="AVERAG
E":L$="62 TO 73    72 TO 78"
1040 IF ANT<-5 AND ANT>-21 THEN A$="BELO
W AVERAGE":L$="60 TO 66    65 TO 71"
1050 IF ANT<-20 THEN A$="POOR":L$="60 OR
 LESS 65 OR LESS"
1060 IF ANT<-45 THEN A$="VERY POOR"
1070 IF ANT<-60 THEN A$="VERY, VERY POOR
"
1080 IF ANT>5 AND ANT<15 THEN A$="GOOD":
L$="74 TO 80    79 TO 85"
1090 IF ANT>14 THEN A$="EXTREMELY GOOD":
L$="81 PLUS     86 PLUS"
1100 PRINT "THIS INDICATES YOUR HEALTH S
TATUS       IS ";A$
1110 PRINT
1120 PRINT "LIFE EXPECTANCY:"
1130 PRINT TAB(3);"MALE          FEMALE"
1140 PRINT TAB(3);L$
1150 END
1160 REM ************
```

```
1170 REM ACCEPT INPUT
1180 IF INKEY$<>"" THEN 1180
1190 A$=INKEY$
1200 IF A$<"A" OR A$>"E" THEN 1190
1210 PRINT:PRINT TAB(12);"OK     ";A$
1220 RETURN
1230 REM ***************
1240 REM DELAY/SPACE OUT
1250 FOR J=1 TO 1000:NEXT J
1260 CLS
1270 PRINT:PRINT:PRINT:PRINT
1280 PRINT "WHICH OF THE FOLLOWING IS CL
OSEST         TO THE TRUTH (SELECT ONE):"
1290 PRINT
1300 FOR J=1 TO 400:NEXT J
1310 RETURN
```

The expertise in MEDICI comes from very rough approximations I drew from Dr Thomas Holmes' famous *Social Readjustment Rating Scale;* longevity charts published by the Metropolitan Life Insurance Company; Dr Donald Vickey's book *Life Plan for your Health* (Addison-Wesley, Reading, MA., 1978) and Dyveke Spino's fascinating book *New Age Training for Fitness and Health* (Grove House Inc., San Francisco, CA., 1979). As I am not a doctor, and as I've made some very rough and ready approximations in working out this program (including lots of running it over and over again, and fiddling with the results), its output should be taken with a moderately large grain of salt. However, it is not totally devoid of realism, and serves well as a demonstration of what a diagnostic expert system could be like.

Chapter Six
Growing Your Own Expert

Although programs such as THE AUTO MECHANIC and MEDICI are of interest, and demonstrate clearly how many expert systems currently in use in the world work, they have two major disadvantages. The first is that they only encode expertise on a specific subject; they are domain-bound. The second disadvantage is that before the program can be written, someone has to actually have the expertise and know the answers in order to construct the knowledge base.

If you already know how to check for minor problems with your car, you're not likely to find it worthwhile turning on your computer to discover the state of your spark plugs.

However, as I mentioned earlier in the book, many 'real world' expert systems were developed only after the knowledge of a human expert was drawn out and put into some sort of systematic framework by a 'knowledge engineer'. The engineer tries to track down the heuristics by which a problem is solved by the human expert. (A heuristic is a path towards a goal which has been worked out by experience, rather than by calculation; it is a working rule-of-thumb. A path of this type is not guaranteed to produce a certain result, although experience has shown that in the majority of cases it is likely to at least come close to achieving a particular goal. Chess programs play, to a large extent, heuristically. An *algorithmic* approach, in contrast to a heuristic one, is a one in which a technique or procedure is applied which inevitably produces a particular result. Your computer uses inbuilt algorithms for such things as adding numbers together.)

The organised expertise from a human expert is preserved within a program, so the expertise can be tapped. Think how much more interesting and valuable it would be if we could create a program which was able to deduce a set of rules for itself, simply from raw data. This would be an achievement. It would be a bigger achievement if we could write a program which could do this in a field in which even human experts lack precise rules.

If it was possible to do this — to create a program which could create its own rules from raw data, and use them to get satisfactory results in ways in which humans could not — it would be extremely valuable if the programe could then explain to use the nature of the rules it has developed, so we could apply them ourselves. As well as being a servant, the expert system would then have become a research assistant and an instructor.

We are going to write such a program in this section of the book. Although staticians may be a little horrified at the somewhat unrigorous way in which some of the numbers are manipulated by the program, the proof of the expert system is in the performance. If it can become an expert on just about anything, even in fields where human beings are unable to create reliable heuristics, it really doesn't matter too much what the numbers are doing.

Meet FUZZY RITA

Our system has another feature. Many decisions which face real-world experts do not have black and white outcomes. Although the answer to the question "Is number A bigger than number B" can only have one answer, a query like "Is vitamin X good for you" moves us quickly into the world of "it depends". "Vitamin X is often good for new-born babies, except when they have ES-type blood; if their mothers practice karate this can modify the value of X to such babies." This 'rule of thumb' includes *often good, except when* and *can modify;* all pretty loose, fuzzy concepts.

As you're sure to have guessed from the name, FUZZY RITA is at home in a world in which cause and effect can not always be put in neat little categories. (The name *Rita* comes from the highly-successful London West End comedy *Educating Rita.* Our program can be educated, as the subroutine from line 970 indicates. The adjective *Fuzzy* not only makes for a delightful name but indicates the nature of the reasoning the program can demonstrate. It will also perform well in a black and white world of yes/no answers.)

The Components of an Expert System

Before we proceed with the development of our system, we will review what we discussed earlier regarding the major parts of expert systems in general. Most commercially available programs have two major components, an *inference engine* and a *knowledge base.* The engine is the mechanism by which the expert system reaches conclusions. The knowledge base, fairly obviously, is the hard data which the system manipulates in order to reach its conclusions.

There is one big advantage in keeping the two parts of the system separate. If the inference mechanism is a general-purpose device, it should be able to work with knowledge bases in various domains, 'thinking' in any field determined by the nature of the knowledge base.

We saw how this could work, earlier in the book, when we discussed MYCIN, the powerful Standford system for diagnosing infectious diseases. Stripped of its knowledge base, it became the inference engine EMYCIN (for Essential MYCIN). The addition of new knowledge bases transformed the doctor first into an auto mechanic (with the aid of a 1975 *Pontiac Service Manual),* a structural analysis consultant (SACON), a medical specialist in the field of bleeding problems (CLOT), as well as systems with such intriguing names as LITHO, HEADMED and BLUEBOX.

FUZZY RITA is not in the same family as MYCIN and its offspring. Rather than having to be spoon-fed a mass of rules, FUZZY RITA develops its own rules, modifying them when necessary if its conclusions are wrong. As it is fascinating to watch the rule base being built up, FUZZY RITA allows you the option of viewing the current state of the rules after each decision has been made.

A General System

FUZZY RITA was specifically created to be a *general* system, able to cope with data on almost any problem. This means that it is not always as 'clever' as it could be. As well, it can sometimes be a little long-winded in requesting information before reaching a conclusion (although it does have the ability, when more than two conclusions are possible, of determining that certain questions cannot effect the outcome; in such cases it will simply skip over these questions).

Therefore, if you intend to use RITA in a specific field, for a long period of time, it is worth 'tweaking' it a bit, once you have it up and running, in order to improve its performance. The weight the system gives to exact matches between user answers and the information in its acquired knowledge base, and the attention it gives to those answers which are not exactly right, may well need a little fiddling in order to get the best results. All expert systems are tested by feeding them known examples, and comparing the system's answer with the known data. Wrong answers suggest the rule base needs modifying, and you should not be afraid to do such modifying.

Despite this, you'll find RITA works surprisingly well in most situations. I worked out the limits which determine how RITA decides on the 'most likely' and 'next most likely' conclusions simply by trial and error, subjecting the system to a wide range of examples, and changing the values as I went along.

Is It a Cat or a Dog?

This is one of the questions which have plagued wise people since the beginning of recorded history. Now, with the help of FUZZY RITA, you can determine it once and for all.

From following through with RITA in action on this contrived example, it should be easy to understand how RITA works, and how it can be used on your own problems. The CAT/DOG problem is one in which the answer is 'discrete'; it is *either* a cat or a dog. Medical diagnoses may well often be non-discrete: *The evidence indicates an infection of the upper respiratory tract, or it could be a mild form of measles.* FUZZY RITA can deal with both discrete and non-discrete outcomes, and does not need to be told which field the current problem occupies. The answers are always given in the form of THE MOST LIKELY RESULT IS . . . If the data which RITA has been fed, in conjunction with the rule base it has created, suggests that there are one or two almost equally likely results, it can give one or two additional conclusions prefaced by the words THE NEXT MOST LIKELY RESULT IS . . . Once RITA has learned the difference between a cat and a dog, you'll generally get a single conclusion.

The program begins by asking you if you want to see the updated knowledge base after each run. Press any key, before RETURN, if you do:

```
PRESS ANY KEY, THEN <RETURN> IF YOU
WANT TO SEE THE UPDATED KNOWLEDGE BASE
      AFTER EACH RUN; JUST PRESS
        <RETURN> IF YOU DON'T
? L
```

Next you enter the 'output options' (that is, the possible conclusions the system can reach), pressing RETURN when you have finished entering these. FUZZY RITA, as written, can cope with up to fifty different outcomes:

```
----------------------------------------
ENTER OUTPUT OPTION NUMBER 1
    (PRESS <RETURN> TO END)
? DOG
----------------------------------------
ENTER OUTPUT OPTION NUMBER 2
    (PRESS <RETURN> TO END)
? CAT
----------------------------------------
```

Once you've finished entering the options, RITA prints them up on the screen, and asks you to enter the 'questions' which the system will later ask. RITA can cope with up to fifty questions:

```
DOG
CAT
----------------------------------------
    ENTER QUESTION NUMBER 1
    (PRESS <RETURN> TO END)
? EATS MICE AND GOES MIAOW
----------------------------------------
    ENTER QUESTION NUMBER 2
    (PRESS <RETURN> TO END)
? ANIMAL
----------------------------------------
    ENTER QUESTION NUMBER 3
    (PRESS <RETURN> TO END)
? BARKS FIERCELY AT ROBBERS
----------------------------------------
```

Again, you press RETURN to signal that you have finished entering questions. Once this has been done, RITA will tell you what subjects it is (or soon will be) capable of deciding between:

```
HERE ARE THE THE SUBJECTS I CAN
DISCRIMINATE BETWEEN:

 >  DOG
 >  CAT
-----------------------------------------
THINK OF ONE, THEN PRESS <RETURN>
```

The questions are then presented one by one, and you are asked to respond to them with a number between 1 (meaning 100% true) and 0 (for 100% false). This allows RITA to cope with discrete data (of the CAT/DOG, BLACK/WHITE variety) and with data which covers a range of values (such as the weight, or systolic blood pressure, of a patient).

With data which can cover a wide range (like weight), it is easy to devise a system which allows you to represent the lowest weight reading as zero, and the highest as one. You'll see, in one weather forecasting example later, I simply divided the maximum and minimum temperatures by 10, meaning the vast majority of the temperatures were entered simply as a number such as .6 (for 6 degrees). Any result above 1 I simply left as 1. Hours of sunshine were treated the same way, with 7 hours being entered as .7, and so on. It does not really matter what the numbers are, so long as you invent a rule which allows the range to fall between 1 and 0, and you stick to this rule, for that item of data. You can have different rules for different categories of data within a single program; RITA does not mind.

You can also use the number you enter to represent such concepts as 'very', 'always', and 'not important' in answer to such questions as

HOW RED IS THE PATIENT'S FACE (where presumably .9 means 'very', and .1 indicates 'not very much at all); DO YOU OFTEN HAVE PROBLEMS STARTING THE CAR (entering, say, .9 for 'always', and .4 for 'from time to time'); and HOW IMPORTANT IS THE MORTGAGE RATE TO THIS LOAN APPLICANT (1 for 'the most important issue', through to .7 for 'fairly important' and .1 for 'not important').

The entered numbers do not have to be exact. Indeed, it hard to see how you can get a consistent numerical equivalent for a concept like 'hardly ever'. With RITA, close enough is nearly always good enough, so long as you are reasonably consistent within a particular category of data.

As you can see, this means RITA is capable of coping with almost any problem you can throw at it, whether the data is of the YES/NO type, is on a measured scale (like temperature), or is of the 'shades of meaning' variety. Our CAT/DOG example is, at least for our purposes, almost totally a YES/NO situation.

First we teach RITA about dogs:

```
---------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

EATS MICE AND GOES MIAOW? 0
---------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

ANIMAL? 1
---------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]
```

```
BARKS FIERCELY AT ROBBERS? 1

THE MOST LIKELY RESULT IS DOG

IS THE MOST LIKELY RESULT CORRECT
          (Y OR N)? Y
```

By luck, the system comes up with DOG, the correct answer (actually, it wasn't luck, but simply that if RITA has no information at all on which to make a decision, it picks the first output option).

RITA updates the rule base, and then reports its findings to us:

```
----------------------------------------
DOG

EATS MICE AND GOES MIAOW 0
ANIMAL 2
BARKS FIERCELY AT ROBBERS 4

----------------------------------------
CAT

EATS MICE AND GOES MIAOW 0
ANIMAL 0
BARKS FIERCELY AT ROBBERS 0

PRESS <RETURN> TO CONTINUE TRAINING
OR ANY KEY THEN <RETURN> TO USE RITA?
```

It has assigned a value of 0 to the question EAST MICE AND GOES MIAOW, 2 to ANIMAL and 4 to BARKS FIERCELY AT ROBBERS. This does not mean it somehow thinks the BARKS

FIERCELY is more important than the ANIMAL category, as it has no information upon which to make such a decision. RITA multiplies the value it gives to each answer by a number which doubles with each question, so it multiples the answer to the first question by 1, the second by 2, the third by 4, the fourth by 8 . . . and so on. You'll see that, so far, the CAT rule base is still full of zeros. This is because RITA has not yet come across a cat, and knows nothing about them.

Let's present the program with a cat, and see what it makes of it:

```
--------------------------------------
THINK OF ONE, THEN PRESS <RETURN>
?
--------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

EATS MICE AND GOES MIAOW? 1
--------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

ANIMAL? 1
--------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

BARKS FIERCELY AT ROBBERS? 0

THE MOST LIKELY RESULT IS DOG

IS THE MOST LIKELY RESULT CORRECT
        (Y OR N)? N
```

The system comes up with DOG, as it is the best match to the data it holds (the ANIMAL question gained the same answer). Once you tell it that DOG was wrong, RITA checks to see how many alternative outcomes it has. If there are only two, it knows the outcome it did not give is the correct one for the data presented during the run, and adjusts its rule base accordingly:

```
-------------------------------------
DOG

EATS MICE AND GOES MIAOW 0
ANIMAL 2
BARKS FIERCELY AT ROBBERS 4

-------------------------------------
CAT

EATS MICE AND GOES MIAOW 1
ANIMAL 2
BARKS FIERCELY AT ROBBERS 0

PRESS <RETURN> TO CONTINUE TRAINING
OR ANY KEY THEN <RETURN> TO USE RITA?
```

You can see that RITA now has created the rule that a positive answer to EATS MICE AND GOES MIAOW and ANIMAL equals CAT, while positive answers to ANIMAL and BARKS FIERCELY AT ROBBERS equals dog. From now on, in this simplest possible case (two outcomes, YES/NO questions) RITA will not make a mistake. It has taught itself to distinguish between two animals.

Not So Straightforward

The situation is not so straightforward when there are more than two outcomes, and RITA may need several trial runs before it is confident that it knows what is going on. Even then, it will ask if the most likely result it has given is correct (and if you say no, will ask about the second most likely one) and will modify its rules slightly if it makes an error. However, when there are more than two outcomes, RITA has an extra skill which comes into play. It can decide that certain questions are irrelevant, and do not help it reach its conclusions. (In the CAT/DOG example, the ANIMAL question was irrelevant; it did not give RITA additional information upon which to decide what sort of creature it was being faced with.)

With the aid of an elementary chemistry text book (White, 1979), I decided to teach RITA to distinguish between magnesium, iron and lead. This, in fact, even for non-experts like myself is not very difficult, but it is considerably different from telling the difference between a cat and a dog, a subject on which I have a 100% expertise rating.

We start by giving RITA the output options:

```
----------------------------------------
ENTER OUTPUT OPTION NUMBER 1
     (PRESS <RETURN> TO END)
? MAGNESIUM
----------------------------------------
ENTER OUTPUT OPTION NUMBER 2
     (PRESS <RETURN> TO END)
? IRON
----------------------------------------
ENTER OUTPUT OPTION NUMBER 3
     (PRESS <RETURN> TO END)
? LEAD
----------------------------------------
```

Then we enter the discrimination questions:

```
----------------------------------------
     ENTER QUESTION NUMBER 1
     (PRESS <RETURN> TO END)
? IS ITS DENSITY BELOW 8 G/CM^3
----------------------------------------
     ENTER QUESTION NUMBER 2
     (PRESS <RETURN> TO END)
? IS IT A METAL
----------------------------------------
     ENTER QUESTION NUMBER 3
     (PRESS <RETURN> TO END)
? IS IT POISONOUS
----------------------------------------
     ENTER QUESTION NUMBER 4
     (PRESS <RETURN> TO END)
? DOES IT CONDUCT ELECTRICITY
----------------------------------------
     ENTER QUESTION NUMBER 5
     (PRESS <RETURN> TO END)
? IS IT SHINY WHEN POLISHED
----------------------------------------
     ENTER QUESTION NUMBER 6
     (PRESS <RETURN> TO END)
? DOES IT TARNISH EASILY
```

RITA reports on the starting situation, and the training begins with lead as our chosen metal:

```
HERE ARE THE THE SUBJECTS I CAN
DISCRIMINATE BETWEEN:

  >  MAGNESIUM
  >  IRON
  >  LEAD
```

```
----------------------------------------
THINK OF ONE, THEN PRESS <RETURN>
?
----------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]

IS ITS DENSITY BELOW 8 G/CM^3? 0
----------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]

IS IT A METAL? 1
----------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]

IS IT POISONOUS? 1
----------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]

DOES IT CONDUCT ELECTRICITY? 1
----------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]

IS IT SHINY WHEN POLISHED? 0
----------------------------------------
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]

DOES IT TARNISH EASILY? 1
```

```
THE MOST LIKELY RESULT IS MAGNESIUM
THE NEXT MOST LIKELY IS LEAD

IS THE MOST LIKELY RESULT CORRECT
          (Y OR N)? N
IS MY SECOND CHOICE CORRECT
    (Y OR N)? Y
```

We run through the questions a few more times, until the knowledge base looks like this:

```
----------------------------------------

MAGNESIUM

IS ITS DENSITY BELOW 8 G/CM^3 1
IS IT A METAL 2
IS IT POISONOUS 0
DOES IT CONDUCT ELECTRICITY 8
IS IT SHINY WHEN POLISHED 0
DOES IT TARNISH EASILY 32

----------------------------------------

IRON

IS ITS DENSITY BELOW 8 G/CM^3 1
IS IT A METAL 2
IS IT POISONOUS 0
DOES IT CONDUCT ELECTRICITY 8
IS IT SHINY WHEN POLISHED 16
DOES IT TARNISH EASILY 32
```

```
----------------------------------------
LEAD

IS ITS DENSITY BELOW 8 G/CM^3 0
IS IT A METAL 2
IS IT POISONOUS 4
DOES IT CONDUCT ELECTRICITY 8
IS IT SHINY WHEN POLISHED 0
DOES IT TARNISH EASILY 32
```

This time, before pressing RETURN for the next run, we press another key, to indicate that training days are over, and it is time for RITA to begin work:

```
PRESS <RETURN> TO CONTINUE TRAINING
OR ANY KEY THEN <RETURN> TO USE RITA? F
```

One practical result of switching from training to working, apart from a slight change in the 'conversation', is that RITA now decides which questions will be asked, rather than automatically asking all of them:

```
HERE ARE THE THE SUBJECTS I CAN
DISCRIMINATE BETWEEN:

  >  MAGNESIUM
  >  IRON
  >  LEAD
----------------------------------------
I AM READY NOW TO DETERMINE WHICH ONE
YOU HAVE
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)   [$ TO END RUN]
```

```
IS ITS DENSITY BELOW 8 G/CM^3? 1
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

IS IT POISONOUS? 0
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

IS IT SHINY WHEN POLISHED? 1

THE MOST LIKELY RESULT IS IRON
THE NEXT MOST LIKELY IS MAGNESIUM

IS THE MOST LIKELY RESULT CORRECT
        (Y OR N)? Y
```

From now on, the knowledge base will only be updated if RITA is told that one of its conclusions is wrong.

```
IS ITS DENSITY BELOW 8 G/CM^3? 0
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

IS IT POISONOUS? 1
 ENTER A NUMBER FROM
1 (TRUE) TO 0 (FALSE)  [$ TO END RUN]

IS IT SHINY WHEN POLISHED? 0

THE MOST LIKELY RESULT IS LEAD
THE NEXT MOST LIKELY IS MAGNESIUM

IS THE MOST LIKELY RESULT CORRECT
        (Y OR N)? Y
```

Chapter Seven
Fuzzy Reasoning

While it is reasonably easy to encode certainties (IF THIS is true THEN THIS IS ALWAYS true) into a computer program, expressing *degrees* of certainty is not so simple. Fuzzy logic, a term introduced by L A Zadeh (1979), deals with reaching conclusions from premises which lack precision.

The conclusions drawn are expressed in terms of *possibilities* rather than *probabilities* such as are possible when the degree of certainty of a premise can be expressed precisely in a mathematical form. Most reasoning mechanisms (such as IF A IS ALWAYS B, AND C IS A THEN C IS B) are unable to cope with imprecision and possibilities.

Imprecision lies behind the majority of human actions, from understanding speech to deciding which move to make in a game of chess. Computers do not, as we have seen, cope well with imprecision. A bit is either 0 or 1 (and never 'possibly 1'). Encoding mechanisms to cope with fuzzy situations demands particular skills from programmers.

In languages like PROLOG and HASTE (see chapters 11 through to 13), the degree of certainty is directly encoded. When using such declarative languages we can use terms such as *in most cases, often, usually, sometimes, hardly ever,* and so on. What is more, the terms can be linked together and compared, so we can say things like *If it is very likely that A is present then often B is hardly ever present.*

In human (i.e. natural) languages, there is a set of terms which are used in our everyday lives which gives degrees of possibility to descriptive terms. The following should make it clear what I mean by that. Think of an adjective, such as *red, long,* or *soggy.* When you describe and object as red, long, or soggy, you often precede the adjective with a word or phrase which modifies the adjective in some way, giving it a more exact meaning. Such words and phrases as *very, only slightly, not particularly* and *not at all* give the adjective a degree of probability. In such cases, we can say that the 'linguistic variable' is made up of two parts, a *primary term* (such as 'red') and a *modifier.*

Although it is clearly impossible to give an exact probability rating for every use of particular modifiers (such as, for example, .95 for *very* and .2 for *not particularly)* it is possible, in practice, to assign values based on our understanding of the degree of 'strength' a particular modifier lends to an adjective, and — more importantly — on our knowledge of the object or whatever which is being descrived and the meaning of the adjective.

Back in the Real World

The CAT/DOG and MAGNESIUM/IRON/LEAD examples with FUZZY RITA were both contrived and demanded only YES/NO answers. It is time to see if the program can work as well with less definite data, and in a situation in which it is not easy (or impossible) to work out what the rules are. Such a situation is weather prediction. Although we can look at the sky in the morning, and say something profound like "It looks like it is going to rain" or "Let's hope it clears up later", we probably have little hard data with which to predict what the rest of today's, or tomorrow's, weather is going to be like.

We are going to look at weather patterns in two very different cities, London and Melbourne. We'll look fairly briefly at the London pattern first, using four variables (hours of sunshine, maximum and minimum temperatures and millimeters of rain). Despite only using these few variables, you'll discover that RITA manages to perform extraordinarily well. Examining RITA's performance in this field will make it easier to understand how the program works when we go through its important segments.

Once we've looked at the London example, and worked our way through the program, we'll give it a set of more precise weather facts related to Melbourne, and see if its performance improves when it is given more exact data.

London

I told RITA that I wanted it to choose from one of three predictions for the one day's weather, given information regarding the preceding day. It was to predict one of the following:

- — Rain tomorrow below 1mm
- — Rain tomorrow above 4mm
- — Rain tomorrow from 1 — 4mm

Its discrimination questions were:

- — Minimum temperature (divided by 10, to make it into a value between zero and one, with any result above 1 entered as one)
- — Maximum temperature (with the figure manipulated in the same way)
- — Rainfall (divided by ten, left as one if over one)

— Hours of sunshine (divided by 10; no answers above one were created by this process; all figures in this exercise were rounded to the first decimal place)

After a week of inputs, and RITA had encountered days which fitted all three possible outcomes, the rule base looked like this:

```
----------------------------------------
RAIN TOMORROW BELOW 1 MM

MINIMUM TEMPERATURE (C  /10) .4
MAXIMUM TEMPERATURE (C  /10) 1.9
RAINFALL (MM  /10) .5
SUNSHINE (HOURS  /10) 3.8

----------------------------------------
RAIN TOMORROW ABOVE 4 MM

MINIMUM TEMPERATURE (C  /10) .3
MAXIMUM TEMPERATURE (C  /10) 1.8
RAINFALL (MM  /10) 1.4
SUNSHINE (HOURS  /10) 35.4

----------------------------------------
RAIN TOMORROW FROM 1 - 4 MM

MINIMUM TEMPERATURE (C  /10) .8
MAXIMUM TEMPERATURE (C  /10) 2
RAINFALL (MM  /10) 3.2
SUNSHINE (HOURS  /10) .8

PRESS <RETURN> TO CONTINUE TRAINING
OR ANY KEY THEN <RETURN> TO USE RITA?
```

It is very interesting to examine this rule base and try and work out what rules RITA has devised, and to see how they compare with the kinds of rough 'rules of thumb' we use when predicting the weather.

For predicting rain below 1mm, the program looks for a value of .4 for the minimum temperature (and almost the same, .3, when looking for rain above 4mm), while a value of .8 is indicated as something worth looking for when trying to predict the 1 to 4mm case. This suggests that RITA believes, after examining just one week of weather, that a high (relatively speaking) minimum temperature is likely to lead to a medium rainfall (i.e. 1 to 4mm) the following day, while a low minimum suggests either very little (or no) rain or a lot (over 4mm) rainfall on the following day. This, at least to me, seems surprising.

There's not much in the values assigned for maximum temperature (1.9, 1.8 and 2) so perhaps, for the month in question, the maximum temperature is not too important. Today's rainfall, by contrast, is considered an important variable. Low rain (a value of .5) suggests little rain the next day, which is in accord with our own feelings about weather occurring with a run of dry days, then wet ones, and so on, rather than every day's weather being a self-contained package, independent of the preceding day. However, rather than saying 'heavy rain today means heavy rain tomorrow', RITA has concluded that a high value for rain today (3.2) suggests medium rain (1 to 4mm) the next day, and a moderate value for today's rain (1.4) suggests heavy rain tomorrow. I guess, on reflection, that also seems reasonable. If most of the rain fell down today, perhaps there is less of it about to fall tomorrow.

Finally, we'll look at hours of sunshine, and see what RITA made of these. The figures here show a wide variation (although you need to keep in mind that they've been multiplied by 16 before being added into the database; the *variation* in a particular category is the important thing, not the *raw* numerical *value)*. To predict a day with more than 4mm of rain, RITA looks for lots of sunshine (a value of 35.4 is in the rule base), while very little sun (.8) suggests 1 to 4mm, and somewhat more (3.8) suggests the following day will be almost or completely dry (below 1mm).

RITA's rule base, at this point, seems to contain the following rules:

— When looking for a fairly wet (above 4mm of rain) day tomorrow, RITA looks for a low minimum temperature, moderate rainfall and quite a bit of sun today.

— To predict a dry day (below 1mm of rain) tomorrow, we can see that RITA examines the input for a low minimum temperature (although slightly higher than for the above 4mm case), very little rain and — perhaps surprisingly — a fairly small amount of sun.

— For the intermediate case (1 to 4mm of rain) tomorrow, RITA looks for a fairly high minimum temperature, a high (compared to the figure for the other two cases) 'rain today' figure, and a very low sunshine figure.

How Well Did It Work?

While it is fairly easy (and extremely interesting) to interpret the rule base which RITA built up in this run, it is of no value whatsoever if it doesn't enable the expert system to actually predict the weather. I ran the program for 19 more days, and recorded its first prediction, the second prediction if it gave one (it will give a second, and perhaps a third, prediction if the data does not point overwhelmingly to one particular conclusion), and whether the first prediction was correct. If it was not, I checked the second prediction.

This was the result (out of 19 days):

— **Correct (first prediction only)**	**— 7 days**
— **Second prediction correct (if first one was wrong)**	**— 7 days**
— **Correct (first or second)**	**— 14 days**
— **Totally wrong (first and second)**	**— 5 days**

This is a fair performance, taking into account the 'most likely' and 'next most likely' findings. Remember, we did not give RITA any rules. We simply entered the data, and said whether the prediction made by the program was right or wrong. RITA made up the rules, and from very few examples managed to at least work out enough about the weather to make reasonable predictions. It is pretty amazing (at least to me) that although the program doesn't 'know' what it is doing, it manages to create a rule which works, to some extent, in 'real life'.

What if we had given it more to work on, or had just asked it to predict a wet or dry day? Would RITA have performed better? We will attempt to answer those questions in due course, with the weather data from Melbourne.

RITA continued to learn during the entire 26-day period for which the data was entered. I decided to run through the period again to see if its performance had improved.

This was the result of its findings over the same 19-day period that it worked on before:

— Correct (first prediction only) — 9 days (up 2)

— Second prediction correct (if first one was wrong)	**— 8 days (up 1)**
— Correct (first or second)	**— 17 days**
— Totally wrong (first and second)	**— 2 days (down 3)**

I was extremely pleased with this improvement. It suggested that if we added more and more examples (rather than working through the same ones over and over again, although that appears to have a degree of merit), RITA would continue to learn. One way of giving RITA a lot of exercise (rather than the extremely tiresome one of typing in lots and lots of data) would be to put the recorded results (that is, the rainfall figures and all) in an array, along with the correct outcome, and just leave RITA to choose days at random for an hour or more, constantly refining its prediction constants.

Modifications

RITA builds up, as we've seen, a number which is stored against each discrimination question for each outcome option. If the system gives a wrong answer, it alters the numbers which it has stored for that option. It multiplies the figure it holds by five, adds in the new figure, and divides by six, and stores the result of this calculation. This ensures that a rogue set of data (such as a cat which barks fiercely at robbers, or a day when one hundred times the normal rainfall fell) does not totally destroy the more usual results. (A rogue set of data for the very first input, however, takes a long, long time for RITA to get over.)

I tried other ratios of new to old data (like nine times the old, plus the new, divided by ten; and double the old, plus the new, divided by three) but they either meant the system learned far too slowly, or its database fluctuated wildly in response to the latest set of figures it had encountered. Like the rest of RITA (and many other expert systems), the program was built up on a trial and error basis. I simply proceeded on a 'change it if it doesn't work/leave it if it does' basis.

Dissecting RITA

I'm now going to work through the RITA program in some detail, a practice I have not followed elsewhere in this book. I am doing it for RITA because this is the most useful program in the book and it is the one you are most likely to want to adapt for creating your own expert systems. As written, it is a general purpose system.

I suggested earlier that you might want to change some of the things the program does, when working in specific domains of your choice, in order to make RITA work more effectively in that particular domain. Run it as it is in your chosen domain, and then play around it to get as many 'right' answers as you can when the outcome is known. It is then far more likely to be correct when you do not know the outcome. And after all, this is the only time when the expert system actually becomes useful, when you have the data and want the program to make some decision on the basis of it.

The program begins by dimensioning a number of arrays:

```
1370 REM ******************************
1380 REM INITIALIZATION
1390 CLS
1400 REM REDUCE ARRAYS IN NEXT LINE IN
         ACCORDANCE WITH YOUR NEEDS
1410 DIM A$(50),B(50,50),C(50),D(50),E$(
50),F(50),G(50)
1420 X$=""
1430 PRINT "PRESS ANY KEY, THEN <RETURN>
 IF YOU"
```

```
1440 PRINT "WANT TO SEE THE UPDATED KNOW
LEDGE BASE"
1450 PRINT "          AFTER EACH RUN; JUST P
RESS"
1460 PRINT "            <RETURN> IF YOU DON'
T"
1470 INPUT U$
1480 CLS
1490 RETURN
```

The 50's in these arrays are far bigger than you're likely to need and use up quite a bit of memory (around 12K on my system, an IBM PC). The A$ array holds the names of the outcomes, and E$ is for the names of the discrimination questions, so you can easily reduce these if you know in advance how many outcomes there will be, and how many questions you will ask. You can change all the other 50's to equal the number of discrimination questions you will ask. Alternatively, you might like to modify the program so that it asks you at the beginning how many outcomes and questions you have, and then dimensions the arrays in accordance with the answer you give. This section of the program also gets a value for U$ (empty, or not) which determines whether or not the program will display the updated database after each run.

From here RITA goes on to accept the names of the possible conclusions it will be able to reach:

```
1160 REM OUTPUT OPTIONS
1170 TT=0
1180 TT=TT+1
1190 GOSUB 1500
1200 PRINT "ENTER OUTPUT OPTION NUMBER"T
T"    (PRESS <RETURN> TO END)"
1210 INPUT A$(TT)
1220 IF A$(TT)="" OR TT=51 THEN TT=TT-1:
RETURN
1230 GOTO 1180
```

And the discrimination questions which will be asked:

```
1250 REM DISCRIMINATION QUESTIONS
1260 CLS
1270 FOR J=1 TO TT
1280 PRINT A$(J)
1290 NEXT J
1300 DQ=0
1310 DQ=DQ+1
1320 GOSUB 1500
1330 PRINT "     ENTER QUESTION NUMBER"DQ
"    (PRESS <RETURN> TO END)"
1340 INPUT E$(DQ)
1350 IF E$(DQ)="" OR DQ=51 THEN DQ=DQ-1:
RETURN
1360 GOTO 1310
```

RITA asks you questions in the section of code from line 140:

```
140 REM QUESTION USER
150 CLS
160 PRINT "HERE ARE THE THE SUBJECTS I C
AN          DISCRIMINATE BETWEEN:"
170 PRINT
180 FOR J=1 TO TT
190 PRINT " >  ";A$(J)
200 NEXT J
210 GOSUB 1500
220 IF X$="" THEN PRINT "THINK OF ONE, T
HEN PRESS <RETURN>"
230 IF X$<>"" THEN PRINT "I AM READY NOW
 TO DETERMINE WHICH ONE   YOU HAVE"
240 IF X$="" THEN INPUT J$
250 ADD=.5
260 FOR J=1 TO DQ
```

```
270 ADD=ADD+ADD
280 GOSUB 1500
290 IF X$<>"" AND TT>2 THEN 390:REM
    CHECK IF QUESTION CAN BE JUMPED
300 PRINT " ENTER A NUMBER FROM"
310 PRINT "1 (TRUE) TO 0 (FALSE)  [$ TO
END RUN]"
320 PRINT:PRINT E$(J);
330 INPUT H$:IF H$="$" THEN PRINT:PRINT
"THANK YOU":PRINT:END
340 C(J)=VAL(H$)
350 C(J)=ADD*C(J)
360 NEXT J
370 RETURN
```

A string is entered (line 330) after the question is printed by the preceding line. If the entered material is a dollar sign, the program terminates (after a polite THANK YOU) with line 330. Otherwise, the VAL of the string is set equal to an element of the C array (line 340) and this is multiplied by the variable ADD in the next line. Before the J loop begins (which accepts the user input), ADD is set equal to .5 and it is added to itself (that is, its value doubles) each time through the loop (making it worth 1, 2, 4, 8, 16 and so on) before C(J) is multiplied by it. Once the program has been through the loop, control returns to a cycle of subroutine calls near the beginning of the program.

Line 290 sends the program to the subroutine from line 390, which checks to see if a question can be missed out:

```
390 REM CHECK IF QUESTION CAN BE
                  JUMPED
400 JUMP=1
410 FOR W=1 TO TT
420 IF ABS(B(W,J)-B(1,J))>.7 THEN JUMP=0
```

```
430 NEXT W
440 IF JUMP=0 THEN 300
450 C(JUMP)=B(W,J)
460 GOTO 360
```

It does this by comparing all the values held in the B array for that question. If they are within .7 of each other, RITA assumes that the information from that question can be safely ignored. This .7 is a figure you may well want to play with.

Once all the questions have been asked, RITA goes to the most important routine, from line 480, where the decision is made and — if necessary — the rule base is updated:

```
480 REM MAKE DECISION
490 FOR J=1 TO TT
500 D(J)=0:E(J)=0:F(J)=0
510 NEXT J
520 ADD=.5
530 FOR J=1 TO TT
540 ADD=ADD+ADD
550 FOR X=1 TO DQ
560 REM PLAY WITH VALUES IN NEXT THREE
      LINES FOR MOST EFFICIENT RESULTS
570 IF C(X)=B(J,X) THEN D(J)=D(J)+1
580 IF ABS(C(X)-B(J,X))<.6*ADD THEN E(J)
=E(J)+.4
590 IF ABS(C(X)-B(J,X))<1.2*ADD THEN F(J
)=F(J)+.1
600 NEXT X
610 NEXT J
620 A1=1:A2=1:A3=1
630 F1=1:F2=1:F3=1
640 FOR J=1 TO TT
```

```
650 IF D(J)>F1 THEN F1=D(J):A1=J
660 IF E(J)>F2 THEN F2=E(J):A2=J
670 IF F(J)>F3 THEN F3=F(J):A3=J
680 NEXT J
690 REM ** ANNOUNCE RESULT **
700 PRINT
710 CFLG=0
720 PRINT "THE MOST LIKELY RESULT IS ";A
$(A1)
730 IF A2<>A1 THEN PRINT "THE NEXT MOST
LIKELY IS ";A$(A2):CFLG=1
740 IF A3<>A2 AND A3<>A1 THEN PRINT "THE
 NEXT MOST LIKELY IS ";A$(A3):CFLG=2
750 PRINT
760 PRINT "IS THE MOST LIKELY RESULT COR
RECT                    (Y OR N)";
770 INPUT F$
780 IF F$<>"Y" AND F$<>"N" THEN 770
790 IF F$="Y" AND X$<>"" THEN RETURN
800 IF F$="Y" THEN 980
810 IF TT=2 AND A1=1 THEN A1=2:GOTO 980
820 IF TT=2 THEN A1=1:GOTO 980
830 IF CFLG=0 THEN 890
840 PRINT "IS MY SECOND CHOICE CORRECT
               (Y OR N)";
850 INPUT F$
860 IF F$="N" THEN 890
870 IF CFLG=1 THEN A1=A2:GOTO 980
880 IF CFLG=2 THEN A1=A3:GOTO 980
890 GOSUB 1500
900 FOR J=1 TO TT
910 PRINT J;"- ";A$(J)
920 NEXT J
930 PRINT
940 PRINT "WHICH IS THE CORRECT ONE";
```

```
950 INPUT A1
960 IF A1<1 OR A1>TT THEN 950
970 REM    ** EDUCATING RITA **
        (UPDATE KNOWLEDGE BASE)
980 FOR J=1 TO DQ
990 IF B(A1,J)<>0 THEN B(A1,J)=(C(J)+5*B
(A1,J))/6
1000 IF B(A1,J)=0 THEN B(A1,J)=C(J)
1010 B(A1,J)=INT(10*B(A1,J))/10
1020 NEXT J
1030 PRINT
1040 IF U$="" THEN RETURN
1050 FOR J=1 TO TT
1060 PRINT:GOSUB 1500
1070 PRINT A$(J)
1080 PRINT
1090 FOR K=1 TO DQ
1100 PRINT E$(K);B(J,K)
1110 NEXT K
1120 NEXT J
1130 PRINT
1140 RETURN
```

The process begins by setting the elements of D, E and F arrays to zero. The D array will hold the 'most likely' results, E is used for the 'next most likely' and F for the 'next most likely' after E. Next, the variable ADD (which was used, you'll recall in the 'question user' routine to multiply the information entered by the user) is set to .5 so it can be used in the next loop.

A pair of nested loops now take control of the program flow. The J loop runs from one to TT (the number of outcomes) with the X loop running from one to DQ (the number of discrimination questions).

The next three lines are the most important ones in the program. This is where RITA makes its decisions. Line 570 looks for an exact match between the entered answer, C(X), and that element of the B array which relates to the possible outcome (the value TT has at that point) and the question which C(X) is the answer to (the value DQ has at that point). If it finds an exact match, D(J) is incremented by one. That is, the chance of the outcome J being the highest value — and therefore being the answer selected by RITA — is increased by one. You may find your system works best if, say, 1.5 or 2 is added at this point.

Line 580 looks for a close match between a value in the database and the entered answer, and if a match within .6 is found, .4 is added to E(J). Note that one is added for a perfect match to D(J), and .4 for a near match to E(J). Line 590 looks for a 'not so near' match, and if it finds it, adds .1 to F(J).

Lines 620 and 630 set the variables A1, A2, A3, F1, F2 and F3 equal to zero, before the J loop which runs from lines 640 to 680 is activated. As it runs through this loop, RITA sets each F variable to the highest value it can find (setting F1 to the highest D(J); F2 to the highest E(J) and the value of the highest F(J) is given to F3). Each time F1, F2 or F3 is changed, A1, A2 or A3 is changed to equal the *number* of that element which triggered the change (that is, it is set equal to the J at that point).This gives RITA a record of which element has been found so far to have the highest value.

Once RITA has been through the loops, A1 is set equal to the outcome which is most likely to be true (because the most matches and/or near matches between the rule base and the user's answers have been given to that outcome).

Now it is time for RITA to announce its conclusion. Line 710 sets a variable called CFLG (for Close result FLaG) to zero. As the A$ array holds the names of the outcomes, A$(A1) is the element of that array which is the name of the most likely outcome. Line 720 announces this conclusion. If A2 is not equal to A1 (that is, the 'next most likely' is not the same as the 'most likely') RITA gives this result, and sets CFLG to 1 (which will be used in a moment). If A3 has a value different from that held by A1 and A3, a 'next most likely' is given, and CFLG is set to 2.

Line 760 asks if RITA's conclusion is correct. If the answer is yes (that is, F$) is set equal to "Y") and X$ is not equal to "" (which happens when RITA moves out of the training mode, and into its working state), the rule base is not modified. If RITA is in work mode, and the answer is right, then the rule base should not be fiddled with. If RITA is still in training, line 800 sends action to the routine from 980 which updates the rule base.

Two Outcomes

If there are only two outcomes, the variable TT will equal 2. The computer gets to line 870 after a "N" has been entered (indicating that its answer was wrong). Therefore, as there are only two outcomes, the other outcome must be correct. If A1 equals 1, then RITA gave — incorrectly — outcome 1 as the right answer. Line 810 changes this so A1 is now set to the correct answer (that is, to 2) before action goes to 980 to update the rule base. Line 880 does the opposite, changing an incorrect 2 into a 1.

If there are more than two outcomes, RITA has a bit more of a problem. It will not be immediately evident which of the remaining outcomes are correct. Line 830 checks the variable CFLG and if it finds it is equal to zero, knows that RITA has not indicated any 'next most likely' results, so goes to the routine from 890 through to 960 which asks the user which answer was correct.

If RITA has assigned some values to A2 and A3 (the 'next most likelies') which are different from the value assigned to A1 (the most likely), the program asks if its 'second choice', A$(A2), is the correct one. If so, the value of CFLG indicates which answer has been given as the second choice (if CFLG equals 1, it is the value of A2; if CFLG equals 2, it's the value of A3) so RITA knows which answer is right, and goes to the routine from 980 armed with this information, in order to update the rule base.

The next routine, from 900 to 960, we have already looked at. This prints all the outcome options on the screen, and asks the user to indicate which one is the right answer.

Now RITA can update the rule base. It knows what the right answer should be (A1 was either selected by it earlier, or A1 has been set equal to the right answer by the user; in any case, A1 now indicates the right answer).

The program goes through the J loop from lines 980 to 1020. When it comes across the relevant portion of the database, B(A1, J), it checks to see if it is equal to zero. It will be equal to zero if no information has yet been recorded at that point (as will always be the case at the beginning of a run). If B(A1, J) is not equal to zero (line 990) the program multiplies the current value held there by five, adds in the value obtained in the current run, then divides the lot by six. This ensures that (a) all the information from previous runs is not swept away by this answer; (b) the impact of the current answer isn't ignored; and (c) an atypical answer does not throw the rule base off too widely (so a single, unusual cat which swims will not destroy the program's ability to recognise as cats animals which have all the other elements of catness, but which do not swim like our rogue cat).

If B(A1, J) does equal zero (line 1000) then this element is set equal to the only answer the system has come across so far relating to this question and outcome, and so it is set equal to A1. Line 1010 gets rid of extraneous decimal places. (Without this line, I found RITA holding some values to six decimal places, which was absurd, given the highly subject nature of some of the original data.)

Finally, in this lengthy — but most important — section of the program, RITA checks in line 1040 to see if U$ equals "". If it does, it means the user indicated at the very start of the run that he or she did not want to see the current status of the rule base printed out. In this case, it returns to the controlling loop at the start of the program for the next series of inputs. If the user has indicated a desire to see the current rule base (and this is, to my mind, the most interesting part of the whole process), the next section prints it out, as you have seen in the sample runs.

Chapter Eight
The Complete FUZZY RITA

Several sections of the RITA program were given in the previous chapter. However, it was not listed in full. This chapter makes good that omission. In the next chapter, we'll run RITA through another set of weather data, and show a way of treating input data so that it falls neatly along the scale from 0 to 1.

Before that, though, here is the listing:

```
10 REM FUZZY RITA
20 GOSUB 1380:REM INITIALIZE
30 GOSUB 1160:REM OUTPUT OPTIONS
40 GOSUB 1250:REM DISCRIMINATION OPTIONS

50 GOSUB 140:REM QUESTION USER
60 GOSUB 480:REM MAKE DECISION AND
                UPDATE KNOWLEDGE BASE
70 PRINT "PRESS <RETURN> TO CONTINUE";
80 IF X$<>"" THEN INPUT I$:GOTO 50
90 PRINT " TRAINING"
100 PRINT "OR ANY KEY THEN <RETURN> TO U
SE RITA";
110 INPUT X$:GOTO 50
120 END
130 REM ****************************
140 REM QUESTION USER
150 CLS
```

```
160 PRINT "HERE ARE THE THE SUBJECTS I C
AN          DISCRIMINATE BETWEEN:"
170 PRINT
180 FOR J=1 TO TT
190 PRINT " >  ";A$(J)
200 NEXT J
210 GOSUB 1500
220 IF X$="" THEN PRINT "THINK OF ONE, T
HEN PRESS <RETURN>"
230 IF X$<>"" THEN PRINT "I AM READY NOW
 TO DETERMINE WHICH ONE   YOU HAVE"
240 IF X$="" THEN INPUT J$
250 ADD=.5
260 FOR J=1 TO DQ
270 ADD=ADD+ADD
280 GOSUB 1500
290 IF X$<>"" AND TT>2 THEN 390:REM
    CHECK IF QUESTION CAN BE JUMPED
300 PRINT " ENTER A NUMBER FROM"
310 PRINT "1 (TRUE) TO 0 (FALSE)  [$ TO
END RUN]"
320 PRINT:PRINT E$(J);
330 INPUT H$:IF H$="$" THEN PRINT:PRINT
"THANK YOU":PRINT:END
340 C(J)=VAL(H$)
350 C(J)=ADD*C(J)
360 NEXT J
370 RETURN
380 REM ****************************
390 REM CHECK IF QUESTION CAN BE
                JUMPED
400 JUMP=1
410 FOR W=1 TO TT
420 IF ABS(B(W,J)-B(1,J))>.7 THEN JUMP=0
```

```
430 NEXT W
440 IF JUMP=0 THEN 300
450 C(JUMP)=B(W,J)
460 GOTO 360
470 REM ******************************
480 REM MAKE DECISION
490 FOR J=1 TO TT
500 D(J)=0:E(J)=0:F(J)=0
510 NEXT J
520 ADD=.5
530 FOR J=1 TO TT
540 ADD=ADD+ADD
550 FOR X=1 TO DQ
560 REM PLAY WITH VALUES IN NEXT THREE
     LINES FOR MOST EFFICIENT RESULTS
570 IF C(X)=B(J,X) THEN D(J)=D(J)+1
580 IF ABS(C(X)-B(J,X))<.6*ADD THEN E(J)
=E(J)+.4
590 IF ABS(C(X)-B(J,X))<1.2*ADD THEN F(J
)=F(J)+.1
600 NEXT X
610 NEXT J
620 A1=1:A2=1:A3=1
630 F1=1:F2=1:F3=1
640 FOR J=1 TO TT
650 IF D(J)>F1 THEN F1=D(J):A1=J
660 IF E(J)>F2 THEN F2=E(J):A2=J
670 IF F(J)>F3 THEN F3=F(J):A3=J
680 NEXT J
690 REM ** ANNOUNCE RESULT **
700 PRINT
710 CFLG=0
720 PRINT "THE MOST LIKELY RESULT IS ";A
$(A1)
```

```
730 IF A2<>A1 THEN PRINT "THE NEXT MOST
LIKELY IS ";A$(A2):CFLG=1
740 IF A3<>A2 AND A3<>A1 THEN PRINT "THE
 NEXT MOST LIKELY IS ";A$(A3):CFLG=2
750 PRINT
760 PRINT "IS THE MOST LIKELY RESULT COR
RECT                    (Y OR N)";
770 INPUT F$
780 IF F$<>"Y" AND F$<>"N" THEN 770
790 IF F$="Y" AND X$<>"" THEN RETURN
800 IF F$="Y" THEN 980
810 IF TT=2 AND A1=1 THEN A1=2:GOTO 980
820 IF TT=2 THEN A1=1:GOTO 980
830 IF CFLG=0 THEN 890
840 PRINT "IS MY SECOND CHOICE CORRECT
                (Y OR N)";
850 INPUT F$
860 IF F$="N" THEN 890
870 IF CFLG=1 THEN A1=A2:GOTO 980
880 IF CFLG=2 THEN A1=A3:GOTO 980
890 GOSUB 1500
900 FOR J=1 TO TT
910 PRINT J;"- ";A$(J)
920 NEXT J
930 PRINT
940 PRINT "WHICH IS THE CORRECT ONE";
950 INPUT A1
960 IF A1<1 OR A1>TT THEN 950
970 REM    ** EDUCATING RITA **
          (UPDATE KNOWLEDGE BASE)
980 FOR J=1 TO DQ
990 IF B(A1,J)<>0 THEN B(A1,J)=(C(J)+5*B
(A1,J))/6
1000 IF B(A1,J)=0 THEN B(A1,J)=C(J)
1010 B(A1,J)=INT(10*B(A1,J))/10
```

```
1020 NEXT J
1030 PRINT
1040 IF U$="" THEN RETURN
1050 FOR J=1 TO TT
1060 PRINT:GOSUB 1500
1070 PRINT A$(J)
1080 PRINT
1090 FOR K=1 TO DQ
1100 PRINT E$(K);B(J,K)
1110 NEXT K
1120 NEXT J
1130 PRINT
1140 RETURN
1150 REM ****************************

1160 REM OUTPUT OPTIONS
1170 TT=0
1180 TT=TT+1
1190 GOSUB 1500
1200 PRINT "ENTER OUTPUT OPTION NUMBER"T
T"     (PRESS <RETURN> TO END)"
1210 INPUT A$(TT)
1220 IF A$(TT)="" OR TT=51 THEN TT=TT-1:
RETURN
1230 GOTO 1180
1240 REM ****************************
1250 REM DISCRIMINATION QUESTIONS
1260 CLS
1270 FOR J=1 TO TT
1280 PRINT A$(J)
1290 NEXT J
1300 DQ=0
1310 DQ=DQ+1
1320 GOSUB 1500
```

```
1330 PRINT "      ENTER QUESTION NUMBER"DQ
"    (PRESS <RETURN> TO END)"
1340 INPUT E$(DQ)
1350 IF E$(DQ)="" OR DQ=51 THEN DQ=DQ-1:
RETURN
1360 GOTO 1310
1370 REM ****************************
1380 REM INITIALIZATION
1390 CLS
1400 REM REDUCE ARRAYS IN NEXT LINE IN
          ACCORDANCE WITH YOUR NEEDS
1410 DIM A$(50),B(50,50),C(50),D(50),E$(
50),F(50),E(50)
1420 X$=""
1430 PRINT "PRESS ANY KEY, THEN <RETURN>
 IF YOU"
1440 PRINT "WANT TO SEE THE UPDATED KNOW
LEDGE BASE"
1450 PRINT "       AFTER EACH RUN; JUST P
RESS"
1460 PRINT "         <RETURN> IF YOU DON'
T"
1470 INPUT U$
1480 CLS
1490 RETURN
1500 PRINT "------------------------------
----------"
1510 RETURN
```

Chapter Nine
The Bureau of Meteorology

In this chapter, we'll get RITA to become an expert on December weather in Melbourne, Australia. I'll explain the steps I followed to create this expert system in some detail. You should then have a pretty good idea of how to use RITA to create real expert systems of your own.

Firstly, the data has to be put into a form which is suitable for RITA to digest. We said that RITA expects inputs in the range zero to one, with zero being *false* and one being *true*, with intermediate values representing degrees of truth. You don't have to stick with zero to one. RITA is extremely tolerant. However, it is simpler to set one standard and stick with it whenever you are developing an expert system from the bare bones of RITA, than to have to try and work out later just what scale you were using.

The raw data we're giving RITA in this exercise comes from the Melbourne weather conditions for December 1984. We'll give the computer the daily barometric reading at 9 am, the minimum and maximum temperatures and the relative humidity at 3 pm. From these figures, RITA has to tell us whether it is going to rain the following day or not.

The Commonwealth Bureau of Meteorology (who provided the figures) pointed out that the month we are studying contained the highest number of rain days (14) since 1976. This is good, as it means that about half the days in the month were wet, rather than very, very few of them being wet as was the case in 1982. A month which was almost totally dry would have given RITA little challenge, and would have proved little.

Here's the raw data for the first few days:

DATE	BAROM	TEMP. MIN.	TEMP. MIN.	RH%	RAIN-FALL
1	1011.6	11.0	25.5	31	0
2	1006.7	12.6	27.6	29	0
3	1012.4	10.8	16.5	52	2.6

You can see that the numbers are very different in size, with the barometric readings around 1000, the temperatures between 10 and 30, the relative humidity percentage presumably running from zero to 100, and the rainfall from zero to infinity. How do we turn these into neat little figures which will lie upon our scale from zero to one?

It is very simple, and your computer will do nearly all the work for you. Enter and run the next program, and I'll explain it to you:

```
10 REM SCALING
20 DIM X(50),Z(50)
30 CLS
40 INPUT "HIGHEST VALUE";A
50 INPUT "LOWEST VALUE";B
60 A=A+.001
70 B=B-.001
80 C=(A-B)/50
90 X(0)=B
100 FOR J=1 TO 50
110 X(J)=X(J-1)+C
120 Z(J)=J/50
```

```
130 PRINT Z(J),X(J)
140 NEXT J
150 DIFF=(X(2)-X(1))/2
160 COUNT=0
170 COUNT=COUNT+1
180 PRINT "ENTER VALUE";COUNT
190 INPUT Q$
200 IF Q$="" THEN END
210 Q=VAL(Q$)
220 IF Q<B OR Q>A THEN 180
230 FOR J=0 TO 50
240 IF ABS(Q-X(J))<DIFF THEN LPRINT COUN
T"-"Z(J)
250 NEXT J
260 GOTO 170
```

You run the program and follow the prompts. You have a list of figures to be entered, such as the barometric ones for our forecasting. The prompt reads HIGHEST VALUE? so you look through the barometric data looking for the biggest number in the list. In my list it is 1018.1, so this is entered into the computer. The next prompt is LOWEST VALUE? and a search for this finds 994.2, which is also entered.

Now the computer prompts you one by one to enter the data you need, giving you the number (the variable COUNT in line 180) of the item in your list, just in case you get lost. You type in the first figure (1011.6) on our list, and line 240 prints out (in this case, to the printer; just leave the L off LPRINT if you want it to appear on your screen) the value .72 which is the equivalent on the scale zero to one, of 1011.6 on the far less convenient scale of 994.2 to 1018.1. You go through the whole month's data, typing in each of the figures and taking note (or allowing the computer to do it for you) of the results, so they can be entered into RITA in due course.

The same process is followed for the minimum temperatures, the maximum ones, and the relative humidity. Now we are just about to start training RITA on the vagaries of Australian weather. First, though, I made a couple of slight changes to the program. You'll recall I said that the RITA program as given was only a raw framework, which could (and should) be modified to give the most effective results in the field in which you want your expertise exercised.

As the input in this program is given to two decimal places, it seemed absurb to have line 1010 stripping it down to one decimal place, and probably losing vital information in the process. To overcome this, line 1010 was changed to the following:

```
1010 B(A1,J)=INT(100*B(A1,J))/100
```

Line 570, which looks for exact matches between the current run's input and the database, was modified to look for close matches, rather than exact ones, as follows:

```
570 IF ABS(C(X)-B(J,X))<.2*ADD THEN D(J)
=D(J)+1
```

Educating RITA

I was now ready to put RITA through its paces. I began by telling it that RAIN TOMORROW and DRY TOMORROW were the outcome options, and decided which discrimination questions would be asked:

```
----------------------------------------
     ENTER QUESTION NUMBER 1
     (PRESS <RETURN> TO END)
? BAROMETER
```

```
----------------------------------------
    ENTER QUESTION NUMBER 2
    (PRESS <RETURN> TO END)
? MINIMUM TEMPERATURE
----------------------------------------
    ENTER QUESTION NUMBER 3
    (PRESS <RETURN> TO END)
? MAXIMUM TEMPERATURE
----------------------------------------
    ENTER QUESTION NUMBER 4
    (PRESS <RETURN> TO END)
? RELATIVE HUMIDITY
```

After four days of input and correction, RITA's rule base looked like this:

```
----------------------------------------
RAIN TOMORROW

BAROMETER .6
MINIMUM TEMPERATURE .7
MAXIMUM TEMPERATURE 2.46
RELATIVE HUMIDITY 2.15

----------------------------------------
DRY TOMORROW

BAROMETER .72
MINIMUM TEMPERATURE .48
MAXIMUM TEMPERATURE 2.88
RELATIVE HUMIDITY 1.12
```

The most noticeable rule RITA has created related to relative humidity. If it is low, then the next day was likely to be dry. This seemed a very reasonable rule. RITA also appeared to think low minimum temperatures and high barometric readings also pointed towards a dry day.

I ran the program for four more days worth of data. After this, RITA's rule base had changed to the following:

```
----------------------------------------
RAIN TOMORROW

BAROMETER .6
MINIMUM TEMPERATURE .7
MAXIMUM TEMPERATURE 2.46
RELATIVE HUMIDITY 2.15

----------------------------------------
DRY TOMORROW

BAROMETER .84
MINIMUM TEMPERATURE .55
MAXIMUM TEMPERATURE 2.12
RELATIVE HUMIDITY 2.36
```

Although RITA has stuck with its previous opinions on barometer readings and minimum temperatures, it has changed its mind completely on relative humidity. Patiently, I trudged on, entering more data. By the end of month, RITA had settled on this rule base:

```
----------------------------------------
RAIN TOMORROW

BAROMETER .43
MINIMUM TEMPERATURE 1
```

```
MAXIMUM TEMPERATURE 2.23
RELATIVE HUMIDITY 3.74

---------------------------------------

DRY TOMORROW

BAROMETER .75
MINIMUM TEMPERATURE .95
MAXIMUM TEMPERATURE 2.86
RELATIVE HUMIDITY 2.15
```

A low barometric reading, a high minimum temperature and a low maximum one, along with a high relative humidity suggested RAIN TOMORROW, while the opposite conditions pointed to DRY TOMORROW. These ideas did not seem unreasonable. But how did they work in practice?

In the month in question, ignoring day one (as RITA's answer to that is based entirely on the order in which the outcome options were entered), there were 29 days we could check. RITA predicted the presence or absence of rain correctly on 18 of those days, which seemed pretty good. This impression was strengthened by examining the figures, which showed RITA predicted a day on which only 0.2mm of rain fell would be dry, and that a few wet days in the middle of a spell of dry ones were called correctly.

To see if RITA would continue to learn, I ran through the month again. At the end of the second run, RITA had evolved this rule base:

```
---------------------------------------

RAIN TOMORROW

BAROMETER .43
MINIMUM TEMPERATURE 1.01
```

```
MAXIMUM TEMPERATURE 2.13
RELATIVE HUMIDITY 3.97

----------------------------------------

DRY TOMORROW

BAROMETER .75
MINIMUM TEMPERATURE .98
MAXIMUM TEMPERATURE 2.58
RELATIVE HUMIDITY 2.22
```

You can see that RITA has basically reinforced its earlier position, adding a little more to its high value for relative humidity when looking for a wet day. The second month's performance was an improvement on the first run. The same 29 days produced 19 correct predictions, although it still said the 0.2mm day would be dry. the program also correctly predicted that day two of the month would be dry (a feat the first run had no way of doing), bringing the second run's sucess rate to 20 out of 30.

Here is how RITA performed, with an X indicating an error:

Day	Weather	Predicted First run		Predicted Second run	
2	DRY	-		DRY	
3	WET	DRY	X	WET	
4	WET	DRY	X	WET	
5	WET	WET		WET	
6	DRY	DRY		WET	X
7	DRY	WET	X	DRY	
8	DRY	DRY		DRY	
9	DRY	DRY		DRY	
10	DRY	DRY		DRY	

11	WET	DRY	X	DRY	X
12	WET	WET		WET	
13	DRY	WET	X	WET	X
14	WET	WET		WET	
15	WET	DRY	X	DRY	X
16	WET	WET		WET	
17	DRY	DRY		DRY	
18	WET	DRY	X	DRY	X
19	DRY	WET	X	WET	X
20	WET	DRY	X	DRY	X
21	WET	DRY	X	DRY	X
22	DRY	DRY		DRY	
23	DRY	DRY		DRY	
24	DRY	DRY		DRY	
25	WET	WET		DRY	X
26	WET	WET		WET	
27	WET	DRY	X	DRY	X
28	DRY	DRY		DRY	
29	DRY	DRY		DRY	
30	DRY	DRY		DRY	

A chart like this would be very useful if you were trying to produce a real expert system, as it highlights where errors were made. For example, in both runs the days from the 18th to the 21st were called wrongly. It would be worth finding out what the values were at that point, so see if a little massaging of a few parts of the discrimination process was called for.

Now, you might like to get some day by day statistics of your own town, or capital city, and see how well RITA peforms on them.

New Inputs

If you like, you can modify the user response section of the program so that instead of entering a number from zero to one, the user simply selects a word from a menu, which is then translated internally into a suitable number for the system.

Such a menu (with the number which RITA creates from the answer given after each word) could be as follows:

```
SELECT THE OPTION WHICH IS TRUE
FOR THIS QUESTION :

A - ALWAYS                      (1)
B - MOST OF THE TIME            (.8)
C - ABOUT HALF OF THE TIME      (.6)
D - SOME OF THE TIME            (.4)
E - RARELY                      (.2)
F - NEVER                       (0)
```

You may well find this makes your domain-specific RITA not only easier to use, but more effective, as it is likely to extract slightly more consistent answers from the user than it would if an estimate of the truth of the answer to a discrimination question had to be made.

Chapter Ten
Logic and Programming

Getting a machine to behave logically is a vital step along the road to eliciting the kind of behavior from a machine which could be called genuinely intelligent. Those attempting to program logical behaviour into a machine have a long history of the study of logic to draw upon.

Aristotle's famous syllogism. . .

ALL MEN ARE MORTAL

ARISTOTLE IS A MAN

THEREFORE ARISTOTLE IS MORTAL

. . . introduced one fundamental logical concept, 'this conclusions follows from this/these premise(s)'.

Unfortunately, computers are not automatically drawn along this kind of line of thinking by the kind of programming languages in most common use at the moment. Most computer languages in current use, including BASIC, are *imperative.* That is, they are constructed almost completely of commands which are to be obeyed by the computer (LET X=95:LET Y=2*X:PRINT Y). An imperative language is not the best one in which to write programs to mimic logical thinking.

Declarative Languages

For this we need to turn to declarative programming languages. In these, programs are constructed of definitions, which describe relationships between elements the computer is manipulating. When an imperative program is executed, the computer follows a number of orders, making decisions of the IF/THEN type, and then outputs the results of its processing. When executing a declarative program, the computer makes use of the definitions to satisfy a queried link between entered elements. The output of such a program is the link which it discovers.

The majority of computer languages currently in use, such as BASIC and FORTRAN, work very well when the task to be carried out is a 'linear' one, when the approach to the problem demands a policeman (the central processing unit) to direct the 'thinking traffic' down a well-defined path. But such approaches are not suitable for the demands of artificial intelligence and expert systems where a number of elements need to be able to interact simultaneously and freely.

The work being done by such bodies as Japan's Institute for New Generation Computer Technology and the UK's Alvey Programme are drawing away from the straight-line von Neumann path which computer problem-solving has been following since the 1940s. The fifth-generation computer, instead of being a single, sequentially operating processor, looks like being a number of processors working in parallel, each engaged on their separate (but related, and linked) tasks. Each of these tasks is somewhat like a subroutine to a main program, except that instead of being called one by one, and only at particular times in a program's execution, the 'sub-processors' are all grinding away at their work from time to time, constantly 'reporting in' and reacting to the output of the other processors.

The work of the Alvey and Japanese fifth-generation teams has concentrated, in part, on the use of descriptive programming languages such as LISP (LIst Processing) and its derivatives, such as PROLOG (PROgramming in LOGic) and Logo. We will be examining LISP and PROLOG (along with two somewhat simpler other languages — EASLE and HASTE — which I developed as introductions to the use of descriptive or declarative languages) in this section, and by the time you come to the end of it you'll have versions of each language to run on your own computer.

LISP

LISP can be traced back to 1956, when the first major seminar on artificial intelligence was held at Dartmouth College. It was organised by four men, including a young assistant mathematics professor, John McCarthy. The four put forward a proposal to the Rockefeller Foundation that a conference be held on the premise that any features of intelligence could be described with sufficient exactness to enable a machine to simulate it (McCorduck, 1979).

One of the papers at the conference, delivered by Herbert Simon, was about a somewhat inelegant list-processing language he had developed called IPL (Information Processing Language). Chris Bidmead, writing in *Practical Computing* magazine in October, 1984 (p. 129), points out that IPL and the lecture were the seed which eventually gave birth to LISP. "It's (IPL's) low-level pseudo code and assembler-like syntax suggested to him (McCarthy) the idea of an algebraic list-processing language . . ."

McCarthy used Simon's ideas (along with those of several others working in the field, including Alan Newell, J. C. Shaw and IBM's Gelernter) to develop LISP. He has a version up and and running by 1958 (LISP 1) and from it produced LISP 1.5 which is the real forerunner of the majority of LISPs in use today, including (of course) the program SSLISP towards the end of this section of the book.

LISP begins with two data types — atoms and lists. Lists are made up of atoms and/or other lists. LISP does not really use programs as such, but instead evaluates lists. Data and program are thus, to a significant extent, indistinguishable in LISP. You use a LISP program by asking it to scan its list database for a list (or atom) which satisfies certain conditions.

PROLOG

PROLOG, LISP's most vigorous offspring, makes considerable use of the lack of distinction between data and program. To a significant extent, a PROLOG program is made up of a database of lists which can be interrogated.

Alain Colmerauer invented the language in the early 1970s, and it was first implemented in Marseilles in 1972 by Colmerauer and Roussell, as an interpreter written in Algol-W. It was rewritten the following year in Fortran. The new version was considerably more efficient and quickly spread through much of the academic world in Europe and the US. Universities and artificial intelligence researchers gradually developed their own versions of the language in the decade which followed its first implementation. Edinburgh University's DEC-10 PROLOG, which was the first to incorporate a compiler, is generally regarded as the 'standard' implementation of the language.

PROLOG's popularity has increased quite dramatically since the Japanese announced that use of the language would be one of the principle elements in their fifth-generation artificial intelligence project.

micro-PROLOG

Implementations of PROLOG are now available for many microcomputers. The first of these, micro-PROLOG, was written by Frank G McCabe and Keith L Clark, at Imperial College, London, in the Logic Programming Unit. It appeared in 1982, in Z80 assembler for CP/M 2.2 systems. It is now available for many computers including the IBM PC under MS-DOS or CP/M-86. Micro-PROLOG has a much simplified syntax compared with such implementations as the DEC-10 at Edinburgh. However, the language can easily be extended by the user, and is sufficiently powerful to enable useful work to be done with it.

Micro-PROLOG includes a front-end program called SIMPLE, which is more friendly to work with than the LISP-like list form which the program itself uses. In due course, I'll be given you a program which emulates SIMPLE in micro-PROLOG so you can learn a certain amount of the language without having to go to the expense of buying it. Then you'll be able to decide whether or not your interest in PROLOG is strong enough to justify the purchase of a compiler for the language.

A PROLOG program is made up of a database of facts and rules which you can query. To 'break you into' declarative or descriptive languages gently, I've invented a primitive language of my own. You need only enter a relatively short program in order to get this language running on your computer. Skills you gain with this first language — given in the next chapter — can be applied in our version of SIMPLE which we'll be studying in due course.

Chapter Eleven
Thinking in HASTE

My language is called HASTE (for HArtnell's Simple declarative TonguE). You build up a database in HASTE by entering sentences which contain an asterisk, which effectively breaks the sentences into subjects and predicates. The computer accepts these sentences, and from them can answer questions and reach conclusions. This is easy to understand if you look at the following sample run of HASTE. First of all, we tell the computer a number of facts:

```
> JOHN*IS A MAN
> PETER*IS AFRAID OF THE WOLF
> MARY*IS AFRAID OF THE WOLF
> PETER*IS A MAN
> MARY*IS A WOMAN
> PETER*CLIMBS TREES
> MARY*CLIMBS TREES
> A STITCH*IN TIME SAVES NINE
> A PENNY SAVED*IS A PENNY EARNED
> PETER*IS EIGHT FEET TALL
> PETER*IS FOURTEEN YEARS OLD
> PETER*IS A COMPUTER EXPERT
> MARY*IS A COMPUTER EXPERT
> MARY*IS NINE FEET TALL
```

Notice where the asterisk falls within the sentence, directly preceding the verb, and taking the place of the space which would normally appear in that position in the sentence.

To interrogate the database, you enter a question mark (?) once the > prompt appears. If you want to check on whether or not a particular fact is held by HASTE, you simply follow the question mark with the statement you want to check. The program replies with TRUE or FALSE and then the line END OF ANSWER to show that the information it has printed out is all it can give you in response to that particular query.

Here I am checking to see if HASTE has learned about certain subjects:

```
> ?PETER*IS A MAN
TRUE
     > END OF ANSWER <
> ?PETER*IS AFRAID OF THE WOLF
TRUE
     > END OF ANSWER <
```

If you want to know what HASTE knows about a particular subject, you substitute a / in your query line for the information you want it to supply. Here, HASTE reveals what it knows about PETER (as the question, in effect, is 'Give me all the predicates that apply to the subject PETER):

```
> ?PETER*/
IS AFRAID OF THE WOLF
IS A MAN
CLIMBS TREES
IS EIGHT FEET TALL
IS FOURTEEN YEARS OLD
IS A COMPUTER EXPERT
     > END OF ANSWER <
```

You can also supply the predicate part of the statement, and HASTE will supply all the subjects that have that predicate:

```
> ?/*IS AFRAID OF THE WOLF
PETER
MARY
        > END OF ANSWER <
> ?A STITCH*/
IN TIME SAVES NINE
        > END OF ANSWER <
```

I hope you are already getting an idea from this limited language how declarative languages which can be interrogated, can act as expert systems of quite immense power.

More useful than the above forms of questioning is the one in which the computer has to check on the truth of two statements, and supply information which satisfies both those conditions. The next question, which uses an AND, is asking HASTE 'What subject(s) is afraid of the wolf AND is a man?':

```
> ?/*IS AFRAID OF THE WOLF AND /*IS A MA
N
PETER
        > END OF ANSWER <
```

Or, 'What subject(s) climbs trees AND is a computer expert?':

```
> ?/*IS AFRAID OF THE WOLF AND /*IS A CO
MPUTER EXPERT
PETER
MARY
        > END OF ANSWER <
```

For some questions, no answer is the only reply:

```
> ?/*CLIMBS TREES AND /*IS A PENNY EARNE
D
      > END OF ANSWER <
>
```

If HASTE was, let us say, a medical expert system, it could be asked 'What subject (disease) causes red spots and appears in children under the age of four?'. The information the HASTE expert system would use would have been entered in straightforward English (apart from the asterisk). This is one of the real advantages of declarative languages. They allow natural language (within restrictions, of course) input and can reply in a fairly straightforward manner.

If you wish to find out everything which your current HASTE system knows, you enter a / on each side of the asterisk:

```
> ?/*/
JOHN*IS A MAN
PETER*IS AFRAID OF THE WOLF
MARY*IS AFRAID OF THE WOLF
PETER*IS A MAN
MARY*IS A WOMAN
PETER*CLIMBS TREES
MARY*CLIMBS TREES
A STITCH*IN TIME SAVES NINE
A PENNY SAVED*IS A PENNY EARNED
PETER*IS EIGHT FEET TALL
PETER*IS FOURTEEN YEARS OLD
PETER*IS A COMPUTER EXPERT
MARY*IS A COMPUTER EXPERT
MARY*IS NINE FEET TALL
     > END OF ANSWER <
```

Here are the results of a few more queries:

```
> ?MARY*/
IS AFRAID OF THE WOLF
IS A WOMAN
CLIMBS TREES
IS A COMPUTER EXPERT
IS NINE FEET TALL
     > END OF ANSWER <
> ?JOHN*/
IS A MAN
     > END OF ANSWER <
> ?PETER*/
IS AFRAID OF THE WOLF
IS A MAN
CLIMBS TREES
IS EIGHT FEET TALL
IS FOURTEEN YEARS OLD
IS A COMPUTER EXPERT
     > END OF ANSWER <
```

Before I give you the HASTE listing, so you can experiment with its power yourself, here's a summary of the operating rules:

1 — All input is in sentence form, with an asterisk coming between the subject and the predicate of the sentence

2 — You interrogate the database by preceding your input with a question mark

3 — To check if HASTE knows a fact, you enter the fact, preceded by a question mark. It will reply TRUE (it knows it) or FALSE (it doesn't)

4 — A slash (/) is substituted in other queries, in the position within the statement you want the program to answer. This means that ?/*FATHER OF TOM will return something like JOHN IS; ?/*/ will print out the whole database; and ?JOHN IS*/ will return something like FATHER OF TOM

5 — The database can also reply to AND questions, supplying answers for which both statements are true, so ?JOHN/* AND FATHER/* will return all information which is true for both JOHN and FATHER

As you'll see by examining the listing, HASTE works its magic by simply manipulating the elements of a couple of string arrays. You can store up to 255 facts in the database. Here's the listing:

```
10 REM HASTE
20 DIM A$(255),B$(255)
30 F=0:CLS
40 REM **********
50 FLAG=0
60 INPUT "> ",D$
70 IF D$="" THEN END
80 IF LEFT$(D$,1)="?" THEN 200
90 E=0
100 E=E+1
110 IF MID$(D$,E,1)="*" THEN 140
120 IF E<LEN(D$) THEN 100
130 PRINT "INVALID ENTRY":GOTO 50
140 IF FLAG=3 THEN RETURN
150 F=F+1:IF F=256 THEN END
160 A$(F)=LEFT$(D$,E-1)
170 B$(F)=MID$(D$,E+1)
```

```
180 GOTO 50
190 REM **********
200 REM INTERROGATE
210 FLAG=4:TRUE=0
220 IF RIGHT$(D$,1)="/" THEN FLAG=3
230 FOR J=1 TO LEN(D$)-5
240 IF MID$(D$,J,5)=" AND " THEN FLAG=5:
TRUE=J
250 NEXT J
260 IF FLAG=5 THEN 410
270 IF LEFT$(D$,3)="?/*" THEN FLAG=1
280 IF LEFT$(D$,4)="?/*/" THEN FLAG=2
290 IF FLAG=3 THEN GOSUB 90:F$=MID$(D$,2
,E-2)
300 IF FLAG=1 THEN F$=MID$(D$,4)
310 E=0
320 E=E+1
330 IF A$(E)="" AND FLAG=4 AND TRUE=0 TH
EN PRINT "FALSE"
340 IF A$(E)="" THEN 520
350 IF FLAG=4 AND "?"+A$(E)+"*"+B$(E)=D$
 THEN PRINT "TRUE":TRUE=1
360 IF FLAG=3 AND F$=A$(E) THEN PRINT B$
(E)
370 IF FLAG=2 THEN PRINT A$(E);"*";B$(E)

380 IF FLAG=1 AND F$=B$(E) THEN PRINT A$
(E)
390 IF E<255 THEN 320
400 REM **********
410 F$=MID$(D$,4,TRUE-4):G$=MID$(D$,TRUE
+7)
420 E=0
430 E=E+1
440 IF A$(E)="" THEN 520
```

```
450 IF B$(E)=F$ THEN 470
460 IF E<255 THEN 430
470 H=0
480 H=H+1
490 IF B$(H)="" THEN 460
500 IF B$(H)=G$ AND A$(E)=A$(H) THEN PRI
NT A$(E)
510 IF H<255 THEN 480
520 PRINT TAB(5);" > END OF ANSWER <"
530 GOTO 50
```

Chapter Twelve
A Taste of PROLOG

Now that you've had some experience of working with a declarative language, we can move onto PROLOG. The program I'm going to give you will allow your computer to run a restricted version of the PROLOG front-end program, SIMPLE. I've called it PROLOG-A (which stands for PROLOG-Almost).

Although you should be able to learn a fair amount of PROLOG just from reading this next section, and from running the program, it is not intended to be a real tutorial on the language. However, you'll find that if you get a book on PROLOG or micro-PROLOG, you can use the program given here, in conjunction with your book, to learn the basics of operating the SIMPLE front-end of the language.

Before we examine PROLOG-A, here is a sample PROLOG program running on a complete compiler (material within the /*. . .*/ is a comment, like a REM statement in a BASIC program):

```
/* a night/day database */

owl is-a nocturnal
bat is-a nocturnal
cat is-a diurnal
dog is-a diurnal
```

```
/* rules */

x is-a day-sleeper if x is-a nocturnal
x is-a night-sleeper if y is-a diurnal

x fights y if  x is-a day-sleeper
               and y is-a night-sleeper
```

As you can see, we've set up some initial *facts*, using the form is-a. Then, we've given the computer some *rules* which relate those facts. We can now query the computer like this:

```
does(owl is-a day-sleeper)
YES

which((xy): x fights y)
answer is(cat owl)
answer is(cat bat)
answer is(dog owl)
answer is(dog bat)
```

You can see here that the computer is using the rules it has been given to answer questions. This is, fundamentally, what an expert system does. PROLOG can quickly be recognised as a language which seems purpose-built for the construction of expert systems.

Even more usefully, a PROLOG program can be queried to explain how it reaches its conclusions.

```
does(cat fights owl)
YES

why(cat fights owl)
(cat fights owl) shown by
```

```
(cat is-a night-sleeper) shown by
  (cat is-a diurnal) and
  (diurnal is-a night-sleeper)
(owl is-a day-sleeper) shown by
  (owl is-a nocturnal) and
  (nocturnal is a day-sleeper)

(night-sleeper fights day-sleeper) given
```

You can see that PROLOG is far more complex than HASTE, although they are definitely members of the same family of languages. I hope your experiences with HASTE have made it easier for you to understand what is happening in the PROLOG samples you've just looked at.

Now we can look at how PROLOG-A works.

The Martian Way

We start, as we did with HASTE, by entering facts into the database. In PROLOG-A, the prompt is &., with the . indicating that the computer is scanning the keyboard, waiting for input. We get the program to add facts to its database with a very sensible command, ADD. Whereas in HASTE, the entered material was made up of two parts, a subject and a predicate (which included a verb), in PROLOG-A the entered material is in three parts:

1 — a subject (such as BRADLEY)

2 — a relationship (such as IS-THE-FATHER-OF)

3 — another subject or fact (such as ROBERT)

As you can see, from 2 above, the words in each section are linked by hyphens. The sections are separated from each other by spaces. The spaces and hyphens are very important.

Let's tell the program a thing or two:

```
&.ADD(ZAPPA IS-A MARTIAN)

&.ADD(ZERON IS-A MARTIAN)

&.ADD(EATEE IS-A SUPPLEMENT)

&.ADD(DRINKEE IS-A LIQUID)

&.ADD(LIQUIDDO IS-A BEVERAGE)

&.ADD(LASERGUN IS-A WEAPON)

&.ADD(YYPRUS IS-A MARTIAN)
```

We can find out what it knows with the command LIST ALL which simply tells PROLOG to list out all the facts held in its database:

```
&.LIST ALL

ZAPPA IS-A MARTIAN
ZERON IS-A MARTIAN
EATEE IS-A SUPPLEMENT
DRINKEE IS-A LIQUID
LIQUIDDO IS-A BEVERAGE
LASERGUN IS-A WEAPON
YYPRUS IS-A MARTIAN
```

Now we come to the really exciting part of PROLOG, where we see a skill which HASTE never had. This is the ability to combine different inputs by itself to add new information to its database. So far, we've just given it *facts,* which it has dutifully filed away. We will now teach it some *rules* which relate to some of those facts.

Here's the first rule:

```
&.ADD(X IS NOURISHING IF X IS-A BEVERAGE
)
                    COMPILING RULE
> LIQUIDDO IS NOURISHING
```

This tells it that an object (X is a variable in PROLOG) IS NOURISHING if that object IS-A BEVERAGE. The program tells you that it is compiling that rule, and prints out anything it has added to the database as a result of applying that rule to the facts it knows. We give it a second rule:

```
&.ADD(X IS NOURISHING IF X IS-A SUPPLEME
NT)
                    COMPILING RULE
> EATEE IS NOURISHING
```

As I said in the last paragraph, X is a variable in PROLOG. Micro-PROLOG uses X, Y, Z, x, y and z as variables, but we'll be sticking to X and Y. Variables are empty boxes, into which suitable contents can be placed. You *cannot* use a variable name as a name in the database (so we can't call one of our Martians X, or the program could become very confused).

Next we'll ask PROLOG-A to tell us everything it knows which involves the relationship IS (which is, you should note, quite separate from the relationship IS-A):

```
&.LIST IS
LIQUIDDO IS NOURISHING
EATEE IS NOURISHING
```

And so we continue, allowing PROLOG-A to build up its store of facts about the universe in which it will be an expert:

```
&.ADD(X SPEAKS MARTIANESE IF X IS-A MART
IAN)
                        COMPILING RULE
> ZAPPA SPEAKS MARTIANESE
> ZERON SPEAKS MARTIANESE
> YYPRUS SPEAKS MARTIANESE
```

PROLOG also has the ability to compile rules relating to more than one variable as you'll see here:

```
&.ADD(X PROGRAMS-HIS Y IF X SPEAKS MARTI
ANESE AND Y IS-A WEAPON
1.
? )
                          COMPILING RULE
> ZAPPA PROGRAMS-HIS LASERGUN
> ZERON PROGRAMS-HIS LASERGUN
> YYPRUS PROGRAMS-HIS LASERGUN
```

The program here was told that of a relationship (PROGRAMS-HIS) between the variables X and Y which exists if X SPEAKS MARTIANESE *AND* Y IS-A WEAPON.

As we go along, gradually training PROLOG-A to act as an expert system in the demanding domain of Mars and its inhabitants and lifestyle, we can check on what it is learning. Does it know, for example, that laserguns are dangerous?

```
&.IS(LASERGUN IS DANGEROUS)
NO
```

It doesn't, so we decide to teach it:

```
&.ADD(X IS DANGEROUS IF X IS-A WEAPON)
                     COMPILING RULE
> LASERGUN IS DANGEROUS
```

We check to see if the lesson has been learned (using IS to indicate that we are asking a question):

```
&.IS(LASERGUN IS DANGEROUS)
YES
```

We can learn a lot by interrogating the database. We can ask questions in terms of relationships:

```
&.LIST SPEAKS
ZAPPA SPEAKS MARTIANESE
ZERON SPEAKS MARTIANESE
YYPRUS SPEAKS MARTIANESE
```

Or, as with HASTE, we can use variables in the question, making use of the command WHICH:

```
&.WHICH(X : X IS-A MARTIAN)
ZAPPA
ZERON
YYPRUS
NO (MORE) ANSWERS
```

Here we are asking WHICH subject is such that it is true to say that this subject IS-A MARTIAN. Note the spaces in the interrogative form. These are vital if PROLOG-A is to perform satisfactorily. We do not need to restrict ourselves to a single variable:

```
&.WHICH((X Y) : X IS-A Y)
ZAPPA MARTIAN
ZERON MARTIAN
EATEE SUPPLEMENT
DRINKEE LIQUID
LIQUIDDO BEVERAGE
LASERGUN WEAPON
YYPRUS MARTIAN
NO (MORE) ANSWERS
```

This question says, in effect, WHICH two things (X and Y) are there such that they are related by IS-A. Note that PROLOG-A prints out the values of X and Y it finds *without* also printing out the relationship. As well, note that (as in the examples above) answers from PROLOG-A end with the statement NO (MORE) ANSWERS to indicate it has given you all the information it can on that question.

The final skill which PROLOG-A possesses is that of being able to compile rules in which there are two variables. You'll see that PROLOG-A, for all its brains, cannot work with information it does not have:

```
&.ADD(X DRINKS Y IF X SPEAKS MARTIANESE
AND Y IS-A BEVERAGE)
                          COMPILING RULE
> ZAPPA DRINKS LIQUIDDO
> ZERON DRINKS LIQUIDDO
> YYPRUS DRINKS LIQUIDDO

&.ADD(X FEELS-SICK-AFTER-EATING Y IF X S
PEAKS MARTIANESE AND Y IS-A FOOD-CAPSULE
)
                          COMPILING RULE
STATEMENT2 OF INPUT NOT IN DATABASE
```

```
&.LIST IS-A
ZAPPA IS-A MARTIAN
ZERON IS-A MARTIAN
EATEE IS-A SUPPLEMENT
DRINKEE IS-A LIQUID
LIQUIDDO IS-A BEVERAGE
LASERGUN IS-A WEAPON
YYPRUS IS-A MARTIAN

&.ADD(X FEELS-SICK-AFTER-EATING Y IF X S
PEAKS MARTIANESE AND Y IS-A SUPPLEMENT)
                    COMPILING RULE
> ZAPPA FEELS-SICK-AFTER-EATING EATEE
> ZERON FEELS-SICK-AFTER-EATING EATEE
> YYPRUS FEELS-SICK-AFTER-EATING EATEE
```

Happy Families

Entertaining and instructive as that demonstration was, the domain of Mars is, of course, a totally imaginary one, and the relationships were contrived to show PROLOG-A in action. To show that there is more to life than life on Mars, we'll look at a 'real' database, and one which is much more down to earth.

We are going to teach our program the following family tree:

```
MARY JOE ANNE FRED JANE GEORGE ELIZABETH SAM JENNY MAX ROSE TIM
  I____I     I____I     I____I     I__________I     I_____I   I____I
     I          I          I             I              I        I
   JOHN     MICHELLE     LARRY         NICOLA          BILL     GRETA
     I__________I          I_____________I              I_________I
           I                      I                          I    I
       CHRISTINE                GERALD                    BRENDA BERT
           I                      I__________________________I    I
           I                                   I                  I
           I                                  BOB                 I
           I______________________________________________________I
                                  I
                                DAVID
```

We start with the first generation:

```
&.ADD(MARY MOTHER-OF JOHN)

&.ADD(JOE FATHER-OF JOHN)

&.ADD(ANNE MOTHER-OF MICHELLE)

&.ADD(FRED FATHER-OF MICHELLE)
```

```
&.ADD(JANE MOTHER-OF LARRY)

&.ADD(GEORGE FATHER-OF LARRY)

&.ADD(JENNY MOTHER-OF BILL)

&.ADD(MAX FATHER-OF BILL)

&.ADD(ELIZABETH MOTHER-OF NICOLA)

&.ADD(SAM FATHER-OF NICOLA)
```

We'll check to see what PROLOG-A knows so far:

```
&.LIST ALL

MARY MOTHER-OF JOHN
JOE FATHER-OF JOHN
ANNE MOTHER-OF MICHELLE
FRED FATHER-OF MICHELLE
JANE MOTHER-OF LARRY
GEORGE FATHER-OF LARRY
JENNY MOTHER-OF BILL
MAX FATHER-OF BILL
ELIZABETH MOTHER-OF NICOLA
SAM FATHER-OF NICOLA
ROSE MOTHER-OF GRETA
TIM FATHER-OF GRETA
```

We add a few more branches to the family tree database:

```
&.ADD(JOHN FATHER-OF CHRISTINE)

&.ADD(MICHELLE MOTHER-OF CHRISTINE)
```

```
&.ADD(LARRY FATHER-OF GERALD)

&.ADD(NICOLA MOTHER-OF GERALD)

&.ADD(BILL FATHER-OF BRENDA)

&.ADD(GRETA MOTHER-OF BRENDA)

&.ADD(GERALD FATHER-OF BOB)

&.ADD(BRENDA MOTHER-OF BOB)
```

In our Martian domain, the relationships were of the IS-A or DRINKS variety. In this second database, we have FATHER-OF and MOTHER-OF, which means we can ask it, for example, to list out all the MOTHER-OFs it is holding:

```
&.LIST MOTHER-OF
MARY MOTHER-OF JOHN
ANNE MOTHER-OF MICHELLE
JANE MOTHER-OF LARRY
JENNY MOTHER-OF BILL
ELIZABETH MOTHER-OF NICOLA
ROSE MOTHER-OF GRETA
MICHELLE MOTHER-OF CHRISTINE
NICOLA MOTHER-OF GERALD
GRETA MOTHER-OF BRENDA
BRENDA MOTHER-OF BOB
```

We can then check up on the FATHER-OFs:

```
&.LIST FATHER-OF
JOE FATHER-OF JOHN
FRED FATHER-OF MICHELLE
```

```
GEORGE FATHER-OF LARRY
MAX FATHER-OF BILL
SAM FATHER-OF NICOLA
TIM FATHER-OF GRETA
JOHN FATHER-OF CHRISTINE
LARRY FATHER-OF GERALD
BILL FATHER-OF BRENDA
GERALD FATHER-OF BOB
```

The simplest query we can address to PROLOG-A is a query which checks whether a fact is true or not:

```
&.IS(CHRISTINE MOTHER-OF LARRY)
NO

&.IS(LARRY FATHER-OF GERALD)
YES

&.IS(MICHELLE MOTHER-OF GRETA)
NO
```

Note that, as when dealing with Mars, PROLOG-A answers in English with a YES or a NO. Our final question above asked if Michelle was Greta's mother. PROLOG-A told us this was not true. Therefore, it is only reasonable that we should ask it who Greta's mother is:

```
&.WHICH(X : X MOTHER-OF GRETA)
ROSE
NO (MORE) ANSWERS
```

This question, as in some we saw on Mars, is really asking 'Which X is there such that X is the MOTHER-OF GRETA?'. Fathers can get the same treatment:

```
&.WHICH(X : X FATHER-OF JOHN)
JOE
NO (MORE) ANSWERS
```

We can now address a query, using two variables, to find out the name of every MOTHER-OF in the universe of PROLOG-A's expertise:

```
&.WHICH((X Y) : X MOTHER-OF Y)
MARY JOHN
ANNE MICHELLE
JANE LARRY
JENNY BILL
ELIZABETH NICOLA
ROSE GRETA
MICHELLE CHRISTINE
NICOLA GERALD
GRETA BRENDA
BRENDA BOB
NO (MORE) ANSWERS
```

To date, we've been using the relationships MOTHER-OF and FATHER-OF. I think its about time we told PROLOG-A about parents:

```
JOHN PARENT-OF CHRISTINE
LARRY PARENT-OF GERALD
BILL PARENT-OF BRENDA
GERALD PARENT-OF BOB
BILL PARENT-OF BERT
BERT PARENT-OF DAVID
```

But wait. Why should we go to all the trouble of entering PARENT-

OF into our program when we know that someone is a parent if she is a mother. Let our program do the work, via a rule:

```
&.ADD(X PARENT-OF Y IF X MOTHER-OF Y)
                         COMPILING RULE
> MARY PARENT-OF JOHN
                         COMPILING RULE
                         COMPILING RULE
> ANNE PARENT-OF MICHELLE
                         COMPILING RULE
                         COMPILING RULE
> JANE PARENT-OF LARRY
                         COMPILING RULE
                         COMPILING RULE
> JENNY PARENT-OF BILL
                         COMPILING RULE
                         COMPILING RULE
> ELIZABETH PARENT-OF NICOLA
                         COMPILING RULE
```

We feed the program a similar rule about fathers, and then check to see what PROLOG-A has added to its database by use of these rules:

```
&.LIST ALL

MARY MOTHER-OF JOHN
JOE FATHER-OF JOHN
ANNE MOTHER-OF MICHELLE
FRED FATHER-OF MICHELLE
JANE MOTHER-OF LARRY
GEORGE FATHER-OF LARRY
JENNY MOTHER-OF BILL
```

```
MAX FATHER-OF BILL
ELIZABETH MOTHER-OF NICOLA
SAM FATHER-OF NICOLA
ROSE MOTHER-OF GROLA
ROSE MOTHER-OF GRETA
TIM FATHER-OF GRETA
JOHN FATHER-OF CHRISTINE
MICHELLE MOTHER-OF CHRISTINE
LARRY FATHER-OF GERALD
NICOLA MOTHER-OF GERALD
BILL FATHER-OF BRENDA
GRETA MOTHER-OF BRENDA
GERALD FATHER-OF BOB
BRENDA MOTHER-OF BOB
BILL FATHER-OF BERT
GRETA MOTHER-OF BERT
CHRISTINE MOTHER-OF DAVID
BERT FATHER-OF DAVID
JOE PARENT-OF JOHN
FRED PARENT-OF MICHELLE
GEORGE PARENT-OF LARRY
MAX PARENT-OF BILL
SAM PARENT-OF NICOLA
TIM PARENT-OF GRETA
JOHN PARENT-OF CHRISTINE
LARRY PARENT-OF GERALD
BILL PARENT-OF BRENDA
GERALD PARENT-OF BOB
BILL PARENT-OF BERT
BERT PARENT-OF DAVID
MARY PARENT-OF JOHN
ANNE PARENT-OF MICHELLE
JANE PARENT-OF LARRY
JENNY PARENT-OF BILL
ELIZABETH PARENT-OF NICOLA
ROSE PARENT-OF GRETA
```

```
MICHELLE PARENT-OF CHRISTINE
NICOLA PARENT-OF GERALD
GRETA PARENT-OF BRENDA
BRENDA PARENT-OF BOB
GRETA PARENT-OF BERT
CHRISTINE PARENT-OF DAVID
```

Other relationships can be fed in, making our expert very expert indeed in terms of David's family tree:

```
&.ADD(X SIBLING-OF Y IF X MOTHER-OF BOB
AND Y FATHER-OF DAVID)
                        COMPILING RULE
> BRENDA SIBLING-OF BERT
```

We next tell PROLOG-A that MARY IS-MARRIED-TO JOE and then let it into the following secret:

```
&.ADD(X SPOUSE-OF Y IF X IS-MARRIED-TO Y
)
                        COMPILING RULE
                        COMPILING RULE
                        COMPILING RULE
> MARY SPOUSE-OF JOE
```

Our database continues to grow as PROLOG-A's expertise increases:

```
&.ADD(X MARRIED-TO Y IF X FATHER-OF LARR
Y AND Y MOTHER-OF LARRY)
                    COMPILING RULE
> GEORGE MARRIED-TO JANE
```

Our system is only an expert one if we can retrieve information from its database. We can ask short, direct questions about one individual:

```
&.IS(BILL FATHER-OF BRENDA)
YES
```

```
&.WHICH(X : JOE FATHER-OF X)
JOHN
NO (MORE) ANSWERS
```

Or check up on related groups:

```
&.WHICH((X Y) : X FATHER-OF Y)
JOE JOHN
FRED MICHELLE
GEORGE LARRY
MAX BILL
SAM NICOLA
TIM GRETA
JOHN CHRISTINE
LARRY GERALD
BILL BRENDA
GERALD BOB
BILL BERT
```

```
BERT DAVID
NO (MORE) ANSWERS
```

```
&.WHICH((X Y) : X SON-OF Y)
JOHN MARY
JOHN JOE
NO (MORE) ANSWERS
```

Playing the Numbers Game

PROLOG-A has numerical as well as textual skills, as we can see here:

```
&.IS(SUM(12 13 25))
YES
```

Here, we are asking the program if the sum of the first two numbers (12 and 13) is equal to the third number. Here are three more examples of this:

```
&.IS(SUM(12 14 20))
NO

&.IS(SUM(5.4 -8 -3.6))
NO

&.IS(SUM(-7 -9 -16))
YES
```

This is of only limited use. However, SUM can be used to find unknown values. First, we will add two numbers:

```
&.WHICH(X : SUM(5.4 -8 X))
-2.6
NO (MORE) ANSWERS
```

Here is the familiar 'Which X is true such that this X is equal to 12 plus 32?' query form. Alternatively, you can query PROLOG-A in the form of 'Which X, when added to a known number, gives a known result?':

```
&.WHICH(X : SUM(X 45 87))
 42
NO (MORE) ANSWERS
```

Placing the unknown in a different position allows PROLOG-A to do subtraction:

```
&.WHICH(X : SUM(123 X -98))
-221
NO (MORE) ANSWERS
```

This is asking 'Which X is the result of subtracting the second number from the third one?'.

PROLOG-A can also check whether or not a number is an integer:

```
&.IS(37 INT)
YES
```

```
&.IS(-987 INT)
YES

&.IS(88.7 INT)
NO

&.IS(8870 INT)
YES
```

It can be asked to return the integer portion of a floating point number:

```
&.WHICH(X : 1234.8 INT X)
 1234

&.WHICH(X : -8889.2987 INT X)
-8890
```

TIMES is used in much the same way as SUM:

```
&.IS(TIMES(9 3 27))
YES

&.IS(TIMES(27 3 99))
NO

&.WHICH(X : TIMES(12 76 X))
 912
NO (MORE) ANSWERS
```

Just as SUM could be used for subtraction, so TIMES can be used for division. Placing the X at either the first or second position gives the same answer:

```
&.WHICH(X : TIMES(12 X 76))
 6.3333333333333333
NO (MORE) ANSWERS

&.WHICH(X : TIMES(X 12 76))
 6.3333333333333333
NO (MORE) ANSWERS
```

Here, PROLOG-A is answering the question 'Which X is there such that X equals 76 divided by 12?'.

The program can compare the size of numbers (and also the relative position of words, with those coming earlier in alphabetical order being considered 'less than' those which come later):

```
&.IS(123 LESS 97)
NO

&.IS(97 LESS 123)
YES

&.IS(TIM LESS HARTNELL)
NO

&.IS(HARTNELL LESS COMPUTER)
NO

&.IS(EVIL LESS OF-TWO)
YES
```

```
&.IS(A LESS B)
YES

&.IS(AAAA1 LESS A2)
NO

&.IS(AA LESS A1)
NO

&.IS(SHORT LESS SHORTY)
YES
```

You can see from the above that working with mathematics in PROLOG-A is not particularly easy. PROLOG (and LISP) were not really designed for mathematical work, but the provision for some maths was made, and its syntax follows the 'list-processing' syntax of PROLOG's real function in life.

Chapter Thirteen
The PROLOG-A Listing

PROLOG-A is, of course, really a BASIC simulation of PROLOG, rather than a true interpreter, so it doesn't behave in every situation exactly like the SIMPLE front-end of micro-PROLOG. However, it is robust in most situations (although rule compiling is not always as thorough as one would wish). The program will certainly give you a tool with which you can teach yourself PROLOG, and with which you can do a great deal of experimenting.

The listing which follows is pretty formidable. To make it more manageable, I deliberately wrote PROLOG-A so you could leave out all the *mathematical* material without effecting the operation of the program at all (and you can, of course, add the numerical processing in later once you've recovered from entering the slightly-stripped down version of PROLOG-A). To delete the mathematics, simply leave out lines 2430 through to 3260. You could also (but I do *not* advise it) leave out lines 220, 230, 240 and 250. If you do, you'll only have to add them in later when you decide to incorporate the mathematics.

As line 20 points out, all input must be in upper case. The program can cope with up to 1000 lines in its database and can compile up to 100 new facts from a rule (although it is most unlikely this will ever be needed). However, the program runs more and more slowly (as I'd guess you would expect) as the amount of information it is holding grows. This does not really matter, as you'll find its response time to most queries is still only a matter of seconds.

```
10 REM PROLOG-A (SIMPLE FRONT-END)
20 REM * ALL INPUT IN UPPER CASE *
30 GOTO 50
40 PRINT "NO (MORE) ANSWERS":RETURN
50 GOSUB 3270:REM INITIALISE
60 REM ***************
70 PRINT
80 INPUT "&.",J$
90 IF J$="" THEN END
100 IF J$="LIST ALL" THEN GOSUB 1860:GOT
O 70
110 IF LEFT$(J$,5)="LIST " THEN J$=MID$(
J$,5)+" ":GOSUB 990:GOTO 70
120 IF RIGHT$(J$,1)<>")" THEN PRINT "1."
:INPUT M$:J$=J$+M$:GOTO 120
130 LJ=LEN(J$)
140 J$=LEFT$(J$,LJ-1)+" ":REM STRIP FINA
L    ), REPLACE WITH SPACE
150 LJ=LEN(J$)
160 FLAG=0
170 IF LEFT$(J$,4)="ADD(" THEN J$=MID$(J
$,5):FLAG=1
180 RULEFLAG=0:PLUSFLAG=0:ARITHFLAG=0
190 FOR R=1 TO LEN(J$)
200 IF MID$(J$,R,4)=" IF " THEN RULEFLAG
=R:FLAG=6
210 IF MID$(J$,R,5)=" AND " THEN PLUSFLA
G=R
220 IF MID$(J$,R,4)="SUM(" THEN ARITHFLA
G=1
230 IF MID$(J$,R,6)="TIMES(" THEN ARITHF
LAG=2
240 IF MID$(J$,R,6)=" LESS " THEN ARITHF
LAG=3
250 IF MID$(J$,R,3)="INT" THEN ARITHFLAG
=4
```

```
260 NEXT R
270 IF LEFT$(J$,3)="IS(" THEN J$=MID$(J$
,4):FLAG=2
280 IF LEFT$(J$,10)="WHICH(X : " THEN J$
=MID$(J$,11):FLAG=3
290 IF LEFT$(J$,16)="WHICH((X Y) : X " T
HEN J$=MID$(J$,17):FLAG=4
300 IF FLAG=0 THEN PRINT "SYNTAX ERROR":
GOTO 70
310 LJ=LEN(J$)
320 REM NOW SEND TO RELEVANT SUBROUTINES

330 IF PLUSFLAG<>0 THEN GOSUB 1950:GOTO
70:REM ENCODE RULE CONTAINING AND
340 IF RULEFLAG<>0 AND FLAG<>5 THEN GOSU
B 1110:REM ENCODE RULE
350 IF ARITHFLAG<>0 THEN GOSUB 2430:GOTO
 70:REM ARITHMETIC
360 IF RIGHT$(J$,3)=" X " OR RIGHT$(J$,3
)=" Y " THEN J$=LEFT$(J$,LJ-2)+" "
370 LJ=LEN(J$)
380 IF FLAG=1 THEN GOSUB 440:REM ADD
390 IF FLAG=2 THEN GOSUB 520:REM IS
400 IF FLAG=3 THEN GOSUB 610:REM WHICH
410 IF FLAG=4 THEN GOSUB 830:REM WHICH2
420 GOTO 70
430 REM ***************************
440 REM             ADD
450 K=0
460 K=K+1
470 IF Z$(K)="" THEN Z$(K)=J$:RETURN
480 IF K<1000 THEN 460
490 PRINT "MEMORY FULL"
500 RETURN
510 REM ***************************
```

```
520 REM                IS
530 K=0
540 K=K+1
550 IF Z$(K)="" THEN 580
560 IF Z$(K)=J$ THEN PRINT "YES":GOTO 59
0
570 IF K<1000 THEN 540
580 PRINT "NO"
590 RETURN
600 REM *************************
610 REM                WHICH
620 IF LEFT$(J$,1)="X" THEN 710
630 J$=LEFT$(J$,LJ-1)
640 K=0
650 K=K+1
660 IF Z$(K)="" THEN 690
670 IF J$=LEFT$(Z$(K),LEN(J$)) THEN PRIN
T RIGHT$(Z$(K),(LEN(Z$(K))-LEN(J$)))
680 IF K<1000 THEN 650
690 GOSUB 40
700 RETURN
710 REM * QUERY STARTS WITH X *
720 J$=MID$(J$,3,LEN(J$)-3)
730 LJ=LEN(J$)
740 K=0
750 K=K+1
760 IF Z$(K)="" THEN 800
770 Q$=MID$(Z$(K),LEN(Z$(K))-LJ,LJ)
780 IF Q$=J$ THEN PRINT LEFT$(Z$(K),LEN(
Z$(K))-(LJ+2))
790 IF K<1000 THEN 750
800 GOSUB 40
810 RETURN
820 REM *************************
830 REM                WHICH2
```

```
840  J$=LEFT$(J$,LJ-2)
850  LJ=LEN(J$)
860  K=0
870  K=K+1
880  IF Z$(K)="" THEN 960
890  LFLAG=0
900  FOR L=1 TO LEN(Z$(K))-LJ
910  IF MID$(Z$(K),L,LJ)=J$ THEN LFLAG=L
920  NEXT L
930  IF LFLAG=0 THEN 950
940  PRINT LEFT$(Z$(K),LFLAG-2);MID$(Z$(K
),(LFLAG+LJ))
950  IF K<1000 THEN 870
960  GOSUB 40
970  RETURN
980  REM ****************************
990  REM                  LIST
1000  K=0
1010  K=K+1
1020  IF Z$(K)="" THEN RETURN
1030  LFLAG=0
1040  FOR L=1 TO LEN(Z$(K))-LEN(J$)
1050  IF MID$(Z$(K),L,LEN(J$))=J$ THEN LF
LAG=1
1060  NEXT L
1070  IF LFLAG=1 THEN PRINT Z$(K)
1080  IF K<1000 THEN 1010
1090  RETURN
1100  REM ****************************
1110  REM         FORM RULES
1120  R=RULEFLAG
1130  E$=LEFT$(J$,R):F$=MID$(J$,R+4)
1140  IF LEFT$(E$,1)<>"X" THEN PRINT "RUL
E ERROR":GOTO 70
```

```
1150 REM NEXT LINE DETECTS INPUTS LIKE
          X EATS Y IF X IS-A Y
1160 IF RIGHT$(F$,2)="Y " THEN 1390
1170 PRINT TAB(18);"COMPILING RULE"
1180 FOR T=1 TO 100
1190 R$(T)=""
1200 NEXT T
1210 E$=MID$(E$,3):F$=MID$(F$,3)
1220 K=0:RR=0
1230 K=K+1
1240 IF Z$(K)="" THEN 1300
1250 IF RIGHT$(Z$(K),LEN(F$))<>F$ THEN 1
370
1260 RR=RR+1
1270 R$(RR)=LEFT$(Z$(K),(LEN(Z$(K))-LEN(
F$)))+E$
1280 PRINT "> ";R$(RR)
1290 GOTO 1230
1300 IF RR=0 THEN RETURN
1310 RC=0
1320 RC=RC+1
1330 Z$(K)=R$(RC)
1340 IF K<1000 THEN K=K+1
1350 IF RC<RR THEN 1320
1360 RETURN
1370 IF K<1000 THEN 1230
1380 RETURN
1390 REM * RULE WITH 2 VARIABLES *
1400 FOR T=1 TO 100
1410 R$(T)=""
1420 NEXT T
1430 K=0:RR=0
1440 IF K=1000 THEN RETURN
1450 K=K+1
1460 IF Z$(K)="" THEN 1770
```

```
1470 REM SPLIT INTO THREE WORDS
1480 Q$=Z$(K)
1490 J=0
1500 J=J+1
1510 IF MID$(Q$,J,1)=" " THEN 1540
1520 IF J<LEN(Q$) THEN 1500
1530 PRINT "RULE COMPILING ERROR":GOTO 7
0
1540 A$=LEFT$(Q$,J)
1550 Q$=MID$(Q$,J+1)
1560 J=0
1570 J=J+1
1580 IF MID$(Q$,J,1)=" " THEN 1610
1590 IF J<LEN(Q$) THEN 1570
1600 PRINT "RULE COMPILING ERROR":GOTO 7
0
1610 B$=LEFT$(Q$,J)
1620 Q$=MID$(Q$,J+1)
1630 J=0
1640 J=J+1
1650 IF MID$(Q$,J,1)=" " THEN 1680
1660 IF J<LEN(Q$) THEN 1640
1670 PRINT "RULE COMPILING ERROR":GOTO 7
0
1680 PRINT TAB(18);"COMPILING RULE"
1690 C$=LEFT$(Q$,J)
1700 M$=MID$(F$,3,LEN(B$))
1710 IF B$<>M$ THEN 1440
1720 RR=RR+1
1730 N$=MID$(E$,3,LEN(E$)-4)
1740 R$(RR)=A$+N$+C$
1750 PRINT "> ";R$(RR)
1760 GOTO 1440
1770 IF RR=0 THEN RETURN
1780 M=0
```

```
1790 M=M+1
1800 IF M>RR THEN RETURN
1810 Z$(K)=R$(M)
1820 IF K=1000 THEN PRINT "OUT OF MEMORY
":GOTO 70
1830 K=K+1
1840 GOTO 1790
1850 REM ****************************
1860 REM              LIST ALL
1870 PRINT
1880 K=0
1890 K=K+1
1900 IF Z$(K)="" THEN RETURN
1910 PRINT Z$(K)
1920 IF K<1000 THEN 1890
1930 RETURN
1940 REM ****************************
1950 REM FORM RULES WITH 'AND' OF THE
         FOLLOWING TYPE:
1960 REM (X EATS Y IF X IS-A BIRD AND
         Y COMES-IN BOXES)
1970 REM X STATEMENT MUST BE IN LIST
PRECEDING Y FOR ALL EXAMPLES TO BE CODED

1980 REM SPLIT INTO SECTIONS
1990 J$=MID$(J$,2):REM STRIP "X"
2000 PRINT TAB(20);"COMPILING RULE"
2010 J=1
2020 J=J+1

2030 IF MID$(J$,J,1)=" " THEN 2060
2040 IF J<LEN(J$) THEN 2020
2050 PRINT "RULE COMPILING ERROR":RETURN

2060 A$=LEFT$(J$,J):REM RELATIONSHIP 1
```

```
2070 J$=MID$(J$,J+7):REM STRIP
     TO START OF SECOND RELATIONSHIP
2080 J=1:COUNT=0
2090 J=J+1
2100 IF MID$(J$,J,1)=" " THEN COUNT=COUN
T+1
2110 IF COUNT=2 THEN 2140
2120 IF J<LEN(J$) THEN 2090
2130 PRINT "RULE COMPILING ERROR":RETURN

2140 B$=LEFT$(J$,J):REM  STATEMENT1
2150 C$=MID$(J$,J+6):REM STATEMENT2
2160 IF C$=" " THEN PRINT "RULE COMPILIN
G ERROR":RETURN
2170 REM NOW GO THROUGH DATABASE
2180 FOR T=1 TO 200
2190 R$(T)=""
2200 NEXT T
2210 R1=0:R2=99
2220 K=0
2230 K=K+1
2240 IF Z$(K)="" THEN 2310
2250 IF R1=99 OR R2=200 THEN PRINT "MEMO
RY SHORTAGE":GOTO 2310
2260 LB=LEN(B$)
2270 IF RIGHT$(Z$(K),LB)=B$ THEN R1=R1+1
:R$(R1)=LEFT$(Z$(K),LEN(Z$(K))-LB)
2280 LC=LEN(C$)
2290 IF RIGHT$(Z$(K),LC)=C$ THEN R2=R2+1
:R$(R2)=LEFT$(Z$(K),LEN(Z$(K))-LC)
2300 IF K<1000 THEN 2230
2310 IF R$(100)="" THEN PRINT "STATEMENT
2 OF INPUT NOT IN DATABASE":RETURN
2320 REM NOW ENCODE RULES
2330 R1=0:R2=99
```

```
2340 R2=R2+1
2350 R1=R1+1
2360 IF R$(R1)>" " AND R$(R2)>" " THEN Z
$(K)=R$(R1)+A$+R$(R2)+" "
2370 PRINT "> ";Z$(K)
2380 K=K+1
2390 IF R$(R2+1)<>"" THEN 2340
2400 IF R$(R1+1)<>"" THEN 2350
2410 RETURN
2420 REM ***********************
2430 REM          ARITHMETIC
2440 LJ=LEN(J$)
2450 IF ARITHFLAG<3 THEN GOSUB 2490
2460 IF ARITHFLAG=3 THEN GOSUB 2890
2470 IF ARITHFLAG=4 THEN GOSUB 3080
2480 RETURN
2490 REM ***** SUM TIMES *********
2500 J$=MID$(J$,5,LJ-5)
2510 IF LEFT$(J$,2)="S(" THEN J$=MID$(J$
,3)
2520 LJ=LEN(J$)

2530 K=0
2540 K=K+1
2550 IF MID$(J$,K,1)=" " THEN A$=LEFT$(J
$,K-1):J$=MID$(J$,K+1):GOTO 2580
2560 IF K<LJ THEN 2540
2570 PRINT TAB(12);"ARITHMETIC ERROR":RE
TURN
2580 LJ=LEN(J$)
2590 K=0
2600 K=K+1
2610 IF MID$(J$,K,1)=" " THEN B$=LEFT$(J
$,K-1):J$=MID$(J$,K+1):GOTO 2640
2620 IF K<LJ THEN 2600
```

```
2630 PRINT TAB(12);"ARITHMETIC ERROR":RE
TURN
2640 LJ=LEN(J$)
2650 K=0
2660 K=K+1
2670 IF MID$(J$,K,1)=")" THEN C$=LEFT$(J
$,K-1):GOTO 2700
2680 IF K<LJ THEN 2660
2690 PRINT TAB(12);"ERROR (TOO MANY VARI
ABLES)":RETURN
2700 AN=0:BN=0:CN=0
2710 IF ASC(A$)>58 THEN AN=1
2720 IF ASC(B$)>58 THEN BN=2
2730 IF ASC(C$)>58 THEN CN=4
2740 GUIDE=AN+BN+CN:IF GUIDE=3 OR GUIDE=
5 OR GUIDE=6 THEN 2690
2750 IF ARITHFLAG=2 THEN 2820:REM TIMES
2760 IF GUIDE>0 THEN 2790
2770 IF VAL(A$)+VAL(B$)=VAL(C$) THEN PRI
NT "YES":RETURN
2780 PRINT "NO":RETURN
2790 IF GUIDE=1 THEN PRINT VAL(C$)-VAL(B
$):GOSUB 40:RETURN
2800 IF GUIDE=2 THEN PRINT VAL(C$)-VAL(A
$):GOSUB 40:RETURN
2810 PRINT VAL(A$)+VAL(B$):GOSUB 40:RETU
RN
2820 REM * TIMES *
2830 IF GUIDE>0 THEN 2860
2840 IF VAL(A$)*VAL(B$)=VAL(C$) THEN PRI
NT "YES":RETURN
2850 PRINT "NO":RETURN
2860 IF GUIDE=1 THEN PRINT VAL(C$)/VAL(B
$):GOSUB 40:RETURN
```

```
2870 IF GUIDE=2 THEN PRINT VAL(C$)/VAL(A
$):GOSUB 40:RETURN
2880 PRINT VAL(A$)*VAL(B$):GOSUB 40:RETU
RN
2890 REM ******** LESS ***********
2900 NF=0
2910 IF ASC(J$)<58 THEN NF=1:REM NUMBERS

2920 COUNT=0
2930 K=0
2940 K=K+1
2950 IF MID$(J$,K,1)=" " THEN COUNT=COUN
T+1
2960 IF COUNT=2 THEN 3000
2970 IF K<LEN(J$) THEN 2940
2980 PRINT TAB(20);"COMPARISON ERROR"
2990 RETURN
3000 B$=MID$(J$,K+1)
3010 A$=LEFT$(J$,K-6)
3020 IF NF=1 THEN 3050
3030 IF A$<B$ THEN PRINT "YES":RETURN
3040 PRINT "NO":RETURN
3050 REM * NUMBERS *
3060 IF VAL(A$)<VAL(B$) THEN PRINT "YES"
:RETURN
3070 PRINT "NO":RETURN
3080 REM ********* INT ***********
3090 IF RIGHT$(J$,2)="X " THEN 3190
3100 K=0
3110 K=K+1
3120 IF MID$(J$,K,1)=" " THEN 3160
3130 IF K<LEN(J$) THEN 3110
3140 PRINT TAB(20);"ARITHMETIC ERROR"
3150 RETURN
3160 A=VAL(LEFT$(J$,K-1))
3170 IF INT(A)=A THEN PRINT "YES":RETURN
```

```
3180 PRINT "NO":RETURN
3190 K=0
3200 K=K+1
3210 IF MID$(J$,K,1)=" " THEN 3240
3220 IF K<LEN(J$) THEN 3200
3230 PRINT TAB(20);"ARITHMETIC ERROR":RE
TURN
3240 PRINT INT(VAL(LEFT$(J$,K-1)))
3250 RETURN
3260 REM ****************************
3270 REM         INITIALISE
3280 CLS
3290 DIM Z$(1000),R$(200)
3300 RETURN
```

Chapter Fourteen
A Pronounced LISP

PROLOG, which we've studied in the preceding chapters, is an outgrowth of LISP (as are the languages Smalltalk and Logo). LISP is much more difficult to learn to use than is the SIMPLE front-end of PROLOG, but with this increased difficulty of learning comes a significant increase in flexibility and power. Once you've mastered PROLOG-A, you'll be in a fairly good position to dip into LISP.

The Words

There are only two kinds of words in LISP, atoms and lists. An atom is a word (or a combination of letters and numbers, which starts with a letter). It cannot contain spaces or any characters which are not letters or numbers. Here are some atoms:

COMPUTER BOMB R2D2 C3PO

A list is made up of atoms and other lists. It starts with a left parenthesis, followed by atoms and lists, and ends with a right parenthesis. There may well be other pairs of parentheses within the outer pair. The simplest list contains a single atom, like this: (COMPUTER), or more than one atom, separated by spaces: (COMPUTER BOMB R2D2). LISP allows for an empty list, which is just (). It is called the *null list.*

A list can contain other lists. Think of the list (HOW DO YOU DO). It could be *within* another list: (THE QUESTION IS (HOW DO

YOU DO)). Lists (and proliferating parentheses) can get quite complex, as in the list ((HOW (DO YOU (DO IT)) NOW) BRAD). (Indeed, the large numbers of parentheses has given rise to the theory that LISP, instead of being an acronym for LISt Processing, really stands for Lots of Infuriatingly Stupid Parentheses.)

Don't worry if this seems to be becoming complex already. Our LISP-like programs, EASLE and SSLISP, will lead you through your list-manipulations relatively painlessly, doing such things as counting the parentheses for you, and making sure they balance.

The Basic Functions

There are six basic functions in LISP. Our first program, EASLE (Easy And Simple Lisp-like Exerciser), will allow you to experiment with them. The six functions are CAR, CDR, CONS, ATOM, NULL and EQ. On a real LISP system, you can use just these six to create just about any computer function which can be created. You'll soon learn what the six functions do by seeing them in action in EASLE. (Both EASLE and SSLISP follow the syntax of EVQ-LISP, as it is one of the easiest to use. Other dialects, such as EV-LISP or P-LISP, demand slightly different input syntax. However, all LISPs, especially at their basic level, are fairly alike, and the skills you'll gain working with my programs can be fairly easily adapted to the demands of whichever system you end up using.)

CAR

This is used to get the first element of a list. The function is used with a pair of parentheses of its own, enclosing the list you want to check (which is within its own pair of outer parentheses). The simplest case is as follows:

```
> CAR ((THIS IS HEAVY MAN))
  VALUE IS...
  THIS
```

In the next case, the first element of the list is another list, (THIS IS) as you can see when you CAR the list:

```
> CAR (((THIS IS) HEAVY MAN))
   VALUE IS...
   (THIS IS)
```

Where PROLOG answered YES and NO to questions, LISP returns a T (for 'true') or a NIL (for false). An empty list, (), is referred to as NIL by the system, as you'll see here, where the empty list is the first element of the LIST we are CARing:

```
> CAR ((() (THIS IS HEAVY)))
   VALUE IS...
   NIL
```

You'll recall I said earlier that EASLE and SSLISP count parentheses for you. You can see this happening here, when I accidentally left off the final):

```
> CAR ((THIS IS) (THE DAWNING)
 -> MISSING   )
  >    )
   VALUE IS...
   (THIS IS)

>
```

CDR

Pronounced 'cooder'' (or 'cudder' or 'coulder', depending upon which LISP programmer you first meet), CDR returns the list which remains when the CAR has been removed, as you'll see from the following examples:

```
> CDR ((AGE OF AQUARIUS))
  VALUE IS...
  (OF AQUARIUS)

> CDR (((THE AGE OF) AQUARIUS))
  VALUE IS...
  (AQUARIUS)

> CDR (((DAWNING OF THE AGE)))
  VALUE IS...
  NIL

> CDR ((() (AGE OF SPECTRUM)))
  VALUE IS...
  ((AGE OF SPECTRUM))
```

CONS

This function puts lists together (in contrast to CAR and CDR which broke them up). It combines two *arguments,* the second of which must be a list, as should be clear from the next sample run of our EASLE program:

```
> CONS (COMPUTERS (PROVE LITTLE))
  VALUE IS...
  (COMPUTERS PROVE LITTLE)

> CONS ((COMPUTERS) (WORK SLOWLY))
  VALUE IS...
  ((COMPUTERS) WORK SLOWLY)

> CONS ((AGE) ((BEFORE (BEAUTY))))
  VALUE IS...
  ((AGE) (BEFORE (BEAUTY)))
```

```
> CONS (AQUARIUS ())
  VALUE IS...
  (AQUARIUS)
```

You can see that a CONSed list is made up of a CAR and a CDR, the two arguments of the list.

ATOM

From CAR, CDR and CONS we move to ATOM, which is much simpler than the preceding three. You'll remember that I pointed out that, when queried, LISP returns T (for 'true') or NIL (for 'false', or 'empty list'). ATOM (along with the next two functions we'll look at, EQ and NULL) are *test* functions, and returns a T or a NIL. ATOM returns a T if the only item in the list being considered is an atom. The program's operation should make this clear:

```
> ATOM (SPITFIRE)
  VALUE IS...
  T

> ATOM ((AGE OF AQUARIUS))
  VALUE IS...
  NIL

> ATOM ((AQUARIUM))
  VALUE IS...
  NIL
```

EQ

We saw above that ATOM returns a T if the single argument is an atom, and NIL otherwise. EQ (for 'equal') returns T if the two arguments of EQ are identical atoms; otherwise it gives a NIL.

Here is EQ in action:

```
> EQ (AQUARIUS AQUARIUM)
   VALUE IS...
   NIL

> EQ (FISHTANK FISHTANK)
   VALUE IS...
   T
```

NULL

Finally, we look at the sixth function, a test function called NULL. NULL has a single argument, and returns T if the argument is the empty list () and NIL if it is anything else:

```
> NULL (())
   VALUE IS...
   T

> NULL ((()))
   VALUE IS...
   NIL

> NULL ()
ILLEGAL - NULL NEEDS ARGUMENT
   VALUE IS...


> NULL ((DAWNING OF THE AGE))
   VALUE IS...
   NIL
```

EASLE

True LISP (unlike EASLE) allows you to enter elaborate, compound expressions to be evaluated, such as:

```
CONS (CAR ((SAUSAGES)) (CONS (FRY)
               CDR ((NEVER VERY WELL))))
```

This returns:

```
VALUE IS...
(SAUSAGES FRY VERY WELL)
```

SSLISP (which stands for Single Statement LISP) does not allow multiple expressions either, which severely limits its value as a tool in its own right. However, although EASLE and SSLISP cannot be used to create expert systems as they are, the two languages can certainly be used to equip you with the basic skills needed to grapple with a more complete LISP.

Enter and run EASLE now, and use to evaluate several lists of your own:

```
10 REM EASLE
20 CLS
30 DIM X(40),Y(40),Z(40)
40 PRINT
50 INPUT "> ",A$
60 IF A$="" THEN END
70 REM ***************************
80 FOR J=1 TO 40
90 X(J)=0:Y(J)=0:Z(J)=0
100 NEXT J
110 REM ***************************
120 R=0:S=0:T=0:CFIRST=0:CSECND=0:EDGE=0
```

```
130 FOR J=1 TO LEN(A$)
140 B$=MID$(A$,J,1)
150 IF B$="(" THEN S=S+1:Z(S)=J:IF T=0 T
HEN CFIRST=J
160 IF B$=")" THEN T=T+1:Y(T)=J:IF CSECN
D<>0 AND EDGE=0 THEN EDGE=J-1
170 IF T=1 AND B$=")" THEN CSECND=J
180 IF B$=" " THEN R=R+1:X(R)=J
190 NEXT J
200 IF S=T THEN 260:REM ( ) BALANCE
210 IF S<T THEN PRINT " -> MISSING  ("
220 IF S>T THEN PRINT " -> MISSING  )"
230 INPUT "  >   ",B$
240 A$=A$+B$
250 GOTO 80
260 FLAG=0
270 IF LEFT$(A$,5)="CAR (" THEN FLAG=1
280 IF LEFT$(A$,5)="CDR (" THEN FLAG=2
290 IF LEFT$(A$,6)="CONS (" THEN FLAG=3
300 IF LEFT$(A$,6)="ATOM (" THEN FLAG=4
310 IF LEFT$(A$,4)="EQ (" THEN FLAG=5
320 IF LEFT$(A$,6)="NULL (" THEN FLAG=6
330 ON FLAG GOSUB 420,470,550,690,780,92
0
340 IF FLAG<>0 THEN GOSUB 360
350 GOTO 40
360 REM ** RETURN ANSWER **
370 PRINT "   VALUE IS..."
380 IF B$<>"()" THEN PRINT "   ";B$
390 IF B$="()" THEN PRINT "   NIL"
400 RETURN
410 REM ****************************
420 REM        **  CAR  **
430 IF S=2 THEN B$=MID$(A$,Z(2)+1,X(2)-Z
(2))
```

```
440 IF S>2 THEN B$=MID$(A$,CFIRST,CSECND
-CFIRST+1)
450 RETURN
460 REM ****************************
470 REM          **  CDR  **
480 GOSUB 420
490 LB=LEN(B$)+7
500 B$="("+MID$(A$,LB,EDGE-1)
510 IF RIGHT$(B$,2)="))" THEN B$=LEFT$(B
$,LEN(B$)-1)
520 IF MID$(B$,2,1)=" " THEN B$="("+MID$
(B$,3)
530 RETURN
540 REM ****************************
550 REM          **  CONS **
560 B$=MID$(A$,7,LEN(A$)-1)
570 J=0
580 IF LEFT$(B$,1)="(" THEN J=1
590 J=J+1
600 IF MID$(B$,J,1)="(" THEN 630
610 IF J<LEN(B$) THEN 590
620 B$="   > CONS ERROR <":RETURN
630 LB=LEN(B$)-1
640 B$="("+LEFT$(B$,J-1)+MID$(B$,J+1)
650 B$=LEFT$(B$,LB)
660 IF RIGHT$(B$,2)=" )" THEN B$=LEFT$(B
$,LEN(B$)-2)+")"
670 RETURN
680 REM ****************************
690 REM          **  ATOM **
700 A$=MID$(A$,7,LEN(A$)-1)
710 J=0:B$="NIL"
720 J=J+1
730 IF MID$(A$,J,1)=" " OR MID$(A$,J,1)=
"(" THEN RETURN
```

```
740 IF J<LEN(A$) THEN 720
750 B$="T"
760 RETURN
770 REM ******************************
780 REM        **   EQ   **
790 A$=MID$(A$,5)
800 A$=LEFT$(A$,LEN(A$)-1)
810 J=0:B$="NIL"
820 J=J+1
830 IF MID$(A$,J,1)=")" THEN RETURN
840 IF MID$(A$,J,1)=" " THEN 870
850 IF J<LEN(A$) THEN 820
860 RETURN
870 C$=LEFT$(A$,J-1)
880 A$=MID$(A$,J+1)
890 IF C$=A$ THEN B$="T"
900 RETURN
910 REM ******************************
920 REM        **  NULL **
930 B$="NIL"
940 IF A$="NULL ()" THEN PRINT "ILLEGAL
- NULL NEEDS ARGUMENT":B$=""
950 IF A$="NULL (())" THEN B$="T"
960 RETURN
```

Chapter Fifteen SSLISP

From EASLE you have learned how to use the six basic LISP functions, CAR, CDR, CONS, ATOM, EQ and NULL. In our major LISP-like program, SSLISP (for Single Statement LISP), we will have access to the original six, plus many, many of the functions (31 in all) which are supplied with most LISP systems.

SSLISP will even allow you to define your own functions, so if you think FIRST is a more logical name than CAR, you can define a function FIRST which will behave just as CAR does. SSLISP allows you to define up to twenty functions of your own.

As the original six functions are exactly the same in SSLISP as they were in EASLE, I won't give any additional sample runs of them in action. Instead, we will concentrate on the new functions.

MEMBER

MEMBER looks for the presence of the first list within the whole list, and returns the list of which the first list is the CAR. If it cannot find such a list, it returns NIL. MEMBER uses EQUAL (which we will come to in due course) in its definition, wherease the function which follows — MEMQ — uses EQ. (On some systems MEMBER tests for the presence of an atom within a list, returning T or NIL.)

```
: MEMBER ((THE END) (AT (THE END) (WE SI
GN OFF)))
    VALUE IS...
    ((THE END) (WE SIGN OFF))

: MEMBER (THIS ((THIS TEST) PROVES IT))
    VALUE IS...
    NIL
```

MEMQ

As I said above, MEMQ uses EQ in its definition, where MEMBER uses EQUAL. The following sample run shows the effect of this change in practice:

```
: MEMQ (NOW (PERHAPS NOW IS THE TIME))
    VALUE IS...
    (NOW IS THE TIME)

: MEMQ (NOW (PERHAPS (NOW) WE SING))
    VALUE IS...
    NIL
```

APPEND

You can probably guess from its name that APPEND is used to create a single list by joining two other lists. This next sample run shows it in action:

```
: APPEND ((LET US) (JOIN TWO LISTS))
    VALUE IS...
    (LET US JOIN TWO LISTS)
```

```
: APPEND ((IT MAKES TWO) (INTO ONE))
   VALUE IS...
   (IT MAKES TWO INTO ONE)
```

REVERSE

This function is ideal for turning the tables. It reverses the elements of a list, but does *not* reverse elements bound within their own parentheses. This should be clear from the second of our two sample runs of this function:

```
: REVERSE ((UPSIDE DOWN AND BACK))
   VALUE IS...
   (BACK AND DOWN UPSIDE)

: REVERSE ((NINE SAVES (A STITCH (IN TIM
E))))
   VALUE IS...
   ( (A STITCH (IN TIME)) SAVES NINE)
```

EQUAL

We mentioned EQUAL a short time ago when discussing MEMBER and MEMQ. EQUAL tests to see if the two parts of the list it is examining are identical. The two arguments of this function can be, as you can see, numbers, lists, or atoms:

```
: EQUAL ((THIS ONE) (THIS ONE))
   VALUE IS...
   T

: EQUAL (9 9)
   VALUE IS...
   T
```

```
: EQUAL ((THIS (ONE)) (THIS (ONE)))
  VALUE IS...
  T

: EQUAL ((THIS ONE) (THAT ONE))
  VALUE IS...
  NIL

: EQUAL (9 -9)
  VALUE IS...
  NIL
```

LIST

Whereas EQUAL, and many other LISP functions, can take two and only two arguments, LIST will accept any number of them. It returns the values of the arguments of the function (which means it simply returns the LIST):

```
: LIST (WE CAN WORK IT OUT)
  VALUE IS...
  (WE CAN WORK IT OUT)

: LIST (WE CAN (WORK IT) OUT)
  VALUE IS...
  (WE CAN (WORK IT) OUT)

: LIST (WE (CAN (WORK IT)) OUT)
  VALUE IS...
  (WE (CAN (WORK IT)) OUT)

: LIST (WE (CAN (WORK (IT OUT))))
  VALUE IS...
  (WE (CAN (WORK (IT OUT))))
```

Operating Mathematically

We saw the rather strange mathematical contortions demanded by PROLOG earlier in this section of the book. LISP makes you jump through similar hoops.

The first pair of functions we will look at are ADD1 and SUB1, which do just what their names indicate, add or subtract one to the single argument which follows them:

```
: ADD1 (9)
  VALUE IS...
   10

: ADD1 (-100)
  VALUE IS...
  -99

: ADD1 (0)
  VALUE IS...
   1

: SUB1 (0)
  VALUE IS...
  -1

: SUB1 (-99)
  VALUE IS...
  -100
```

ZEROP checks to see if the argument of the function is zero or not, returning either T or N. ONEP does the same as a test for the number one:

```
: ZEROP (1)
  VALUE IS...
  NIL

: ZEROP (9)
  VALUE IS...
  NIL

: ZEROP (0)
  VALUE IS...
  T

: ONEP (1)
  VALUE IS...
  T

: ONEP (0)
  VALUE IS...
  NIL
```

NUMBERP will return a T if the argument is a number, or NIL if it is not, while the function MINUSP checks to see if the argument is a negative number or not:

```
: NUMBERP (NINE)
  VALUE IS...
  NIL

: NUMBERP (9)
  VALUE IS...
  T
```

```
: MINUSP (9)
  VALUE IS...
  NIL

: MINUSP (-9)
  VALUE IS...
  T

:
```

Our next two functions, GREATERP and LESSP, demand two arguments. They compare the value of the first argument with the second, returning T or NIL:

```
: GREATERP (999 12)
  VALUE IS...
  T

: GREATERP (12 888)
  VALUE IS...
  NIL

: LESSP (12 888)
  VALUE IS...
  T

: LESSP (-10 -1)
  VALUE IS...
  T

: LESSP (999 12)
  VALUE IS...
  NIL
```

Next we have MAX and MIN which can accept any number of arguments. They return the maximum and minimum value found in their respective lists:

```
: MAX [9 -8 123 0 33 19]
  VALUE IS...
   123

: MIN [9 -8 123 0 33 19]
  VALUE IS...
  -8
```

MINUS changes the sign of a single argument (effectively multiplying it by minus one):

```
: MINUS [8]
  VALUE IS...
  -8

: MINUS [-8]
  VALUE IS...
   8
```

Operating Numerically

While the above mathematical functions are essentially 'passive', returning a statement about arguments or changing the number by one, the next set are used to actually carry out mathematical operations.

DIFFERENCE returns the difference in value between the two arguments, subtracting the second from the first:

```
: DIFFERENCE [88 12]
  VALUE IS...
   76
```

```
: DIFFERENCE [100 -100]
  VALUE IS...
   200
```

You can raise the first argument to the power of the second with EXPT:

```
: EXPT [3 3]
  VALUE IS...
   27

: EXPT [3 -3]
  VALUE IS...
   3.703704E-02
```

RECIP returns the value of one over the argument:

```
: RECIP [3]
  VALUE IS...
   .3333334

: RECIP [10]
  VALUE IS...
   .1

: RECIP [.1]
  VALUE IS...
   10
```

QUOTIENT divides the first argument by the second:

```
: QUOTIENT (12 3)
  VALUE IS...
   4

: QUOTIENT (3 12)
  VALUE IS...
   .25
```

REMAINDER returns the remainder after a modulo arithmetic division has been performed (in many BASICs, the operator MOD returns the integer value which is the remainder of an integer division: REMAINDER is the LISP equivalent):

```
: REMAINDER (10 3)
  VALUE IS...
   1

: REMAINDER (100 9)
  VALUE IS...
   1

: REMAINDER (99 11)
  VALUE IS...
   0

: REMAINDER (12)
  VALUE IS...
   ERROR - ONLY ONE ARGUMENT

: REMAINDER (12 3)
  VALUE IS...
   0
```

While many of the above functions can only handle two arguments, our final two, PLUS and TIMES, will accept any number of arguments:

```
: PLUS [8 12 -3]
   VALUE IS...
    17

: PLUS [-8 -12 3]
   VALUE IS...
   -17


: TIMES [12 3]
   VALUE IS...
    36

: TIMES [12 3 3]
   VALUE IS...
    108

: TIMES [24 .01]
   VALUE IS...
    .24

: TIMES [24 .1]
   VALUE IS...
    2.4
```

Chapter Sixteen
Defining Your Own LISP Functions

I mentioned, when discussing the six basic functions available on LISP systems (CAR, CDR and the rest), that these could be used to define any functions which could be created on a computer system. LISP uses the function DEFINE to define new functions. Our DEFINE function follows standard LISP syntax for user-defining functions, so the practice you get with SSLISP can be directly transferred to a complete LISP system. However, because SSLISP can only accept a single statement at a time, you are limited here to changing the name of a function which the system supports. This allows you to change the names of up to twenty of the functions provided with SSLISP.

The syntax for using DEFINE is as follows:

```
DEFINE (NEW-NAME (LAMBDA (DUMMY-VAR) (FUNCTION DUMMY-VAR)))
```

In common with just about everything in the world of LISP, the definition of a function is a list. The CAR of the list is the name of the function (NEW-NAME in the sample above). This name can be any atom, but should not be a name which is already in use by your system for a function. The new name is followed by the word LAMBDA (the name coming from the logical formalism called the lambda-calculus, developed and explained by Alonzo Church in his *Introduction to Mathematical Logic,* Vol. 1; Princeton, New Jersey;

Princeton University Press, 1956). Despite its impressive lineage, we can effectively ignore the word, although keep in mind that it has to be there for this (and on just every other LISP system in the world) DEFINE function to work.

After LAMBDA comes a dummy variable (here called DUMMY-VARIABLE of the type upon which you'll expect your defined word to operate). Next is the actual system function you wish NEW-NAME to imitate. (On real LISP systems you can use other definitions in this place, leading to complex and very powerful definitions. FORTH grants you the same power.) The final element in the list is the dummy variable again.

This may all sound a little bewildering. (I also found it one of the more difficult elements of LISP to understand.) However, once you see it in action you're likely to realise it is pretty simple.

The function CAR, as you know by now, returns the *first* element of a list. Our first use of DEFINE will be to create a function called FIRST which will return the CAR of a list. We enter the following into our SSLISP system:

```
: DEFINE (FIRST (LAMBDA (DUMMY) (CAR DUMMY)))
   VALUE IS...
   FIRST
```

We then test it by using FIRST as if it were a standard system function:

```
: FIRST ((THIS IS A TEST))
   VALUE IS...
   THIS
```

```
: FIRST (((THIS IS) HEAVY MAN))
   VALUE IS...
   (THIS IS)
```

It works! If you prefer to use an X, rather than the word TIMES to multiply, you can DEFINE X so this will be so:

```
: DEFINE (X (LAMBDA (NUM) (TIMES NUM)))
   VALUE IS...
   X

: X (8 9)
   VALUE IS...
    72
```

We can use BIGGEST to work as MAX:

```
: DEFINE (BIGGEST (LAMBDA (JJ) (MAX JJ)))
   VALUE IS...
   BIGGEST

: BIGGEST (1 44 -9 182 12)
   VALUE IS...
    182
```

Finally, we'll add BOMB to our vocabulary, where it will perform as ATOM:

```
: DEFINE (BOMB (LAMDA (LIST) (ATOM LIST)))
   VALUE IS...
   BOMB

: BOMB (HAHA)
   VALUE IS...
   T
```

Chapter Seventeen
The SSLISP Listing

Here is the complete listing of SSLISP so you can put your computer through its lisping paces:

```
10 REM S.S. LISP
20 GOTO 3450:REM INITIALISE
30 REM ****************************
40 A$=MID$(A$,E):A$=LEFT$(A$,LEN(A$)-1)
50 RETURN
60 REM ****************************
70 PRINT
80 NN=0
90 INPUT ": ",A$
100 IF A$="" THEN END:REM JUST PRESS
                      <RETURN> TO END RUN
110 REM ****************************
120 FOR J=1 TO 12
130 X(J)=0:Y(J)=0:Z(J)=0
140 NEXT J
150 REM ****************************
160 R=0:S=0:T=0:CFIRST=0:CSECND=0:EDGE=0

170 FOR J=1 TO LEN(A$)
180 B$=MID$(A$,J,1)
190 IF B$="(" THEN S=S+1:Z(S)=J:IF T=0 T
HEN CFIRST=J
200 IF B$=")" THEN T=T+1:Y(T)=J:IF CSECN
D<>0 AND EDGE=0 THEN EDGE=J-1
```

```
210 IF T=1 AND B$=")" THEN CSECND=J
220 IF B$=" " THEN R=R+1:X(R)=J
230 NEXT J
240 IF S=T THEN 300:REM ( ) BALANCE
250 IF S<T THEN PRINT " -> MISSING  ("
260 IF S>T THEN PRINT " -> MISSING  )"
270 INPUT "  +   ",B$
280 A$=A$+B$
290 GOTO 120
300 IF NWDS=0 OR NN=1 THEN 370
310 M$=LEFT$(A$,X(1)-1)
320 FOR J=1 TO NWDS
330 IF M$=N$(J) THEN A$=O$(J)+MID$(A$,LE
N(N$(J))+1)
340 NEXT J
350 NN=1
360 GOTO 120
370 FLAG=0:B$="NIL"
380 IF LEFT$(A$,5)="CAR (" THEN FLAG=1
390 IF LEFT$(A$,5)="CDR (" THEN FLAG=2
400 IF LEFT$(A$,6)="CONS (" THEN FLAG=3
410 IF LEFT$(A$,6)="ATOM (" THEN FLAG=4
420 IF LEFT$(A$,4)="EQ (" THEN FLAG=5
430 IF LEFT$(A$,6)="NULL (" THEN FLAG=6
440 IF LEFT$(A$,8)="MEMBER (" THEN FLAG=
7
450 IF LEFT$(A$,5)="MEMQ " THEN FLAG=8
460 IF LEFT$(A$,8)="APPEND (" THEN FLAG=
9
470 IF LEFT$(A$,9)="REVERSE (" THEN FLAG
=10
480 IF LEFT$(A$,7)="EQUAL (" THEN FLAG=1
1
490 IF LEFT$(A$,6)="LIST (" THEN FLAG=12
```

```
500 IF LEFT$(A$,8)="DEFINE (" THEN FLAG=
13
510 IF LEFT$(A$,6)="ADD1 (" THEN FLAG=14

520 IF LEFT$(A$,6)="SUB1 (" THEN FLAG=15

530 IF LEFT$(A$,7)="ZEROP (" THEN FLAG=1
6
540 IF LEFT$(A$,12)="DIFFERENCE (" THEN
FLAG=17
550 IF LEFT$(A$,6)="EXPT (" THEN FLAG=18

560 IF LEFT$(A$,5)="MAX (" THEN FLAG=19
570 IF LEFT$(A$,5)="MIN (" THEN FLAG=20
580 IF LEFT$(A$,6)="PLUS (" THEN FLAG=21

590 IF LEFT$(A$,7)="MINUS (" THEN FLAG=2
2
600 IF LEFT$(A$,10)="QUOTIENT (" THEN FL
AG=23
610 IF LEFT$(A$,7)="RECIP (" THEN FLAG=2
4
620 IF LEFT$(A$,11)="REMAINDER (" THEN F
LAG=25
630 IF LEFT$(A$,7)="TIMES (" THEN FLAG=2
6
640 IF LEFT$(A$,10)="GREATERP (" THEN FL
AG=27
650 IF LEFT$(A$,7)="LESSP (" THEN FLAG=2
8
660 IF LEFT$(A$,8)="MINUSP (" THEN FLAG=
29
670 IF LEFT$(A$,9)="NUMBERP (" THEN FLAG
=30
680 IF LEFT$(A$,6)="ONEP (" THEN FLAG=31
```

```
690 IF FLAG>13 THEN 720
700 ON FLAG GOSUB 840,890,970,1110,1200,
1330,1380,1510,1700,1760,2000,2130,3300
710 GOTO 760
720 IF FLAG>24 THEN 750
730 ON FLAG-13 GOSUB 2180,2230,2280,2350
,2560,2600,2790,2820,2870,2920,2960
740 GOTO 760
750 ON FLAG-24 GOSUB 3030,3070,3120,3160
,3200,3250,2280
760 IF FLAG<>0 THEN GOSUB 780
770 GOTO 70
780 REM ** RETURN ANSWER **
790 PRINT "   VALUE IS..."
800 IF B$<>"()" THEN PRINT "    ";B$
810 IF B$="()" THEN PRINT "    NIL"
820 RETURN
830 REM ****************************
840 REM          **  CAR  **
850 IF S=2 THEN B$=MID$(A$,Z(2)+1,X(2)-Z
(2))
860 IF S>2 THEN B$=MID$(A$,CFIRST,CSECND
-CFIRST+1)
870 RETURN
880 REM ****************************
890 REM          **  CDR  **
900 GOSUB 840
910 LB=LEN(B$)+7
920 B$="("+MID$(A$,LB,EDGE-1)
930 IF RIGHT$(B$,2)="))" THEN B$=LEFT$(B
$,LEN(B$)-1)
940 IF MID$(B$,2,1)=" " THEN B$="("+MID$
(B$,3)
950 RETURN
```

```
960 REM ****************************
970 REM         **  CONS **
980 B$=MID$(A$,7,LEN(A$)-1)
990 J=0
1000 IF LEFT$(B$,1)="(" THEN J=1
1010 J=J+1
1020 IF MID$(B$,J,1)="(" THEN 1050
1030 IF J<LEN(B$) THEN 1010
1040 B$="  * CONS ERROR *":RETURN
1050 LB=LEN(B$)-1
1060 B$="("+LEFT$(B$,J-1)+MID$(B$,J+1)
1070 B$=LEFT$(B$,LB)
1080 IF RIGHT$(B$,2)=" )" THEN B$=LEFT$(
B$,LEN(B$)-2)+")"
1090 RETURN
1100 REM ****************************
1110 REM        **  ATOM **
1120 A$=MID$(A$,7,LEN(A$)-1)
1130 J=0
1140 J=J+1
1150 IF MID$(A$,J,1)=" " OR MID$(A$,J,1)
="(" THEN RETURN
1160 IF J<LEN(A$) THEN 1140
1170 B$="T"
1180 RETURN
1190 REM ****************************
1200 REM        **   EQ  **
1210 E=5:GOSUB 40
1220 J=0
1230 J=J+1
1240 IF MID$(A$,J,1)=")" THEN RETURN
1250 IF MID$(A$,J,1)=" " THEN 1280
1260 IF J<LEN(A$) THEN 1230
1270 RETURN
1280 C$=LEFT$(A$,J-1)
```

```
1290 A$=MID$(A$,J+1)
1300 IF C$=A$ THEN B$="T"
1310 RETURN
1320 REM ****************************
1330 REM         ** NULL **
1340 IF A$="NULL ()" THEN B$=" * ILLEGAL
 - NULL NEEDS ARGUMENT *"
1350 IF A$="NULL (())" THEN B$="T"
1360 RETURN
1370 REM ****************************
1380 REM         ** MEMBER **
1390 C$=MID$(A$,9)
1400 J=1
1410 J=J+1
1420 IF MID$(C$,J,1)=")" OR MID$(C$,J,1)
="(" THEN D$=LEFT$(C$,J):GOTO 1450
1430 IF J<LEN(C$) THEN 1410
1440 RETURN
1450 J=LEN(D$)
1460 J=J+1
1470 IF MID$(C$,J,LEN(D$))=D$ THEN C$=LE
FT$(C$,LEN(C$)-1):GOTO 1630
1480 IF J<LEN(C$) THEN 1460
1490 RETURN
1500 REM ****************************
1510 REM         ** MEMQ **
1520 C$=MID$(A$,7)
1530 J=0
1540 J=J+1
1550 IF MID$(C$,J,1)=" " THEN 1580
1560 IF J<LEN(A$) THEN 1540
1570 RETURN
1580 D$=LEFT$(C$,J)
1590 C$=MID$(C$,J+2)
1600 C$=LEFT$(C$,LEN(C$)-2)+" "
1610 J=0
```

```
1620 J=J+1
1630 IF MID$(C$,J,LEN(D$))=D$ THEN B$="(
"+MID$(C$,J):GOTO 1660
1640 IF J<LEN(C$) THEN 1620
1650 RETURN
1660 B$=LEFT$(B$,LEN(B$)-1)+")"
1670 IF RIGHT$(B$,3)=")))" THEN B$=LEFT$
(B$,LEN(B$)-1):GOTO 1670
1680 RETURN
1690 REM ******************************
1700 REM         ** APPEND **
1710 B$=MID$(A$,9)
1720 B$=LEFT$(B$,Y(1)-9)+" "+MID$(B$,Z(3
)-7)
1730 B$=LEFT$(B$,LEN(B$)-1)
1740 RETURN
1750 REM ******************************
1760 REM         ** REVERSE **
1770 B$=""
1780 A$=MID$(A$,11):A$=LEFT$(A$,LEN(A$)-
2)
1790 CT=0
1800 J=0
1810 J=J+1:IF J>LEN(A$) THEN 1920
1820 IF MID$(A$,J,1)=" " THEN 1850
1830 IF MID$(A$,J,1)="(" THEN 1860
1840 GOTO 1810
1850 CT=CT+1:G$(CT)=LEFT$(A$,J-1):A$=MID
$(A$,J+1):GOTO 1800
1860 J=J+1:IF MID$(A$,J,2)="))" THEN 198
0
1870 IF MID$(A$,J,1)=")" THEN 1910
1880 IF J=LEN(A$) THEN 1900
1890 GOTO 1860
1900 CT=CT+1:G$(CT)=A$+")":GOTO 1930
```

```
1910 CT=CT+1:G$(CT)=LEFT$(A$,J):A$=MID$(
A$,J+1):GOTO 1800
1920 CT=CT+1:G$(CT)=A$
1930 FOR M=CT TO 1 STEP-1
1940 B$=B$+G$(M):IF M>1 THEN B$=B$+" "
1950 NEXT M
1960 B$="("+B$+")"
1970 RETURN
1980 CT=CT+1:G$(CT)=LEFT$(A$,J+1):A$=MID
$(A$,J+2):GOTO 1800
1990 REM ***************************
2000 REM         ** EQUAL **
2010 E=8:GOSUB 40
2020 M=ASC(A$):IF M>47 AND M<58 THEN 237
0
2030 J=0
2040 J=J+1
2050 IF MID$(A$,J,2)=") " THEN J=J+1:GOT
O 1280
2060 IF MID$(A$,J,3)=")))" THEN 2100
2070 IF MID$(A$,J,2)="))" THEN 2110
2080 IF J<LEN(A$) THEN 2040
2090 RETURN
2100 C$=LEFT$(A$,J+2):A$=MID$(A$,J+4):GO
TO 1300
2110 C$=LEFT$(A$,J+1):A$=MID$(A$,J+3):GO
TO 1300
2120 REM ***************************
2130 REM         ** LIST **
2140 E=7:GOSUB 40
2150 B$="("+A$+")"
2160 RETURN
2170 REM ***************************
2180 REM         ** ADD1 **
2190 E=7:GOSUB 40
2200 B$=STR$(VAL(A$)+1)
```

```
2210 RETURN
2220 REM ****************************
2230 REM        ** SUB1 **
2240 E=7:GOSUB 40
2250 B$=STR$(VAL(A$)-1)
2260 RETURN
2270 REM ****************************
2280 REM        ** ZEROP **
2290 IF FLAG=16 THEN E=8
2300 IF FLAG=31 THEN E=7
2310 GOSUB 40
2320 IF A$="0" AND FLAG=16 OR A$="1" AND
 FLAG=31 THEN B$="T"
2330 RETURN
2340 REM ****************************
2350 REM     ** TWO ARGUMENTS **
2360 E=13:GOSUB 40
2370 J=0
2380 J=J+1
2390 IF MID$(A$,J,1)=" " THEN 2420
2400 IF J<LEN(A$) THEN 2380
2410 B$=" * ERROR - ONLY ONE ARGUMENT *"
:RETURN
2420 P=VAL(LEFT$(A$,J-1))
2430 Q=VAL(MID$(A$,J+1))
2440 IF FLAG=17 THEN B$=STR$(P-Q):RETURN

2450 IF FLAG=23 OR FLAG=25 THEN B=P/Q
2460 IF FLAG=25 THEN B=INT(.5+Q*(B-INT(B
))*1000)/1000
2470 IF FLAG=18 THEN B=P^Q
2480 IF FLAG=11 AND P=Q THEN B$="T"
2490 IF FLAG=27 AND P>Q THEN B$="T"
2500 IF FLAG=28 AND P<Q THEN B$="T"
2510 IF FLAG=32 THEN B=P-Q
```

```
2520 IF FLAG=11 OR FLAG>26 THEN RETURN
2530 B$=STR$(B)
2540 RETURN
2550 REM ****************************
2560 REM        ** EXPT **
2570 E=7:GOSUB 40
2580 GOTO 2370
2590 REM ****************************
2600 REM   ** MAX (MIN PLUS TIMES) **
2610 F$=LEFT$(A$,3):A$=MID$(A$,6)
2620 CT=0:FLAG=0
2630 IF F$="TIMES" THEN CT=1
2640 J=0
2650 J=J+1
2660 IF MID$(A$,J,1)=" " THEN 2690
2670 IF J<LEN(A$) THEN 2650
2680 IF J=LEN(A$) THEN FLAG=1
2690 P=VAL(LEFT$(A$,J-1)):IF FLAG=0 THEN
 A$=MID$(A$,J+1)
2700 IF F$<>"PLUS" AND CT=0 THEN CT=P
2710 IF F$="MAX" AND P>CT THEN CT=P
2720 IF F$="MIN" AND P<CT THEN CT=P
2730 IF F$="PLUS" THEN CT=CT+P
2740 IF F$="TIMES" THEN CT=CT*P
2750 IF FLAG=0 THEN 2640
2760 B$=STR$(CT)
2770 RETURN
2780 REM ****************************
2790 REM        ** MIN **
2800 GOTO 2610
2810 REM ****************************
2820 REM        ** PLUS **
2830 F$="PLUS"
2840 A$=MID$(A$,7)
2850 GOTO 2620
```

```
2860 REM ******************************
2870 REM          ** MINUS **
2880 E=8:GOSUB 40
2890 B$=STR$(-VAL(A$))
2900 RETURN
2910 REM ******************************
2920 REM          ** QUOTIENT **
2930 E=11:GOSUB 40
2940 GOTO 2370
2950 REM ******************************
2960 REM          ** RECIP **
2970 E=8:GOSUB 40
2980 IF A$="0" THEN B$=" DIVISION BY ZER
0 ILLEGAL":RETURN
2990 B=1/(VAL(A$))
3000 B$=STR$(B)
3010 RETURN
3020 REM ******************************
3030 REM          ** REMAINDER **
3040 E=12:GOSUB 40
3050 GOTO 2370
3060 REM ******************************
3070 REM          ** TIMES **
3080 F$="TIMES"
3090 A$=MID$(A$,8)
3100 GOTO 2620
3110 REM ******************************
3120 REM          ** GREATERP **
3130 E=11:GOSUB 40
3140 GOTO 2370
3150 REM ******************************
3160 REM          ** LESSP **
3170 E=8:GOSUB 40
3180 GOTO 2370
3190 REM ******************************
3200 REM          ** MINUSP **
```

```
3210 E=9:GOSUB 40
3220 IF VAL(A$)<0 THEN B$="T"
3230 RETURN
3240 REM *****************************
3250 REM          ** NUMBERP **
3260 A$=MID$(A$,10)
3270 IF ASC(A$)>44 AND ASC(A$)<58 THEN B
$="T"
3280 RETURN
3290 REM *****************************
3300 REM          ** DEFINE **
3310 A$=MID$(A$,9)
3320 F$=LEFT$(A$,X(2)-9)
3330 G$=MID$(A$,X(4)-6)
3340 J=0
3350 J=J+1
3360 IF MID$(G$,J,1)=" " THEN 3390
3370 IF J<LEN(G$) THEN 3350
3380 B$=" DEFINE ERROR":RETURN
3390 G$=LEFT$(G$,J-1)
3400 NWDS=NWDS+1
3410 O$(NWDS)=G$:N$(NWDS)=F$
3420 B$=F$
3430 RETURN
3440 REM *****************************
3450 REM INITIALISE
3460 CLS
3470 DIM G$(20),O$(20),N$(20),X(12),Y(12
),Z(12)
3480 NWDS=0:REM COUNT OF USER-DEFINED
                'NEW WORDS'
3490 GOTO 70
```

Machine-specific listings

Commodore 64

THE AUTO MECHANIC

```
10 REM THE AUTO MECHANIC
15 POKE 53280,0:POKE 53281,0
20 PRINT "
{CLR}"
30 PRINT"SO YOU'RE HAVING AUTOMOBILE PRO
BLEMS."
40 PRINT"LET'S SEE IF WE CAN PIN DOWN
          THE TROUBLE..."
50 PRINT
60 PRINT "ENTER A LETTER WHICH DESCRIBES
 PROBLEM:"
70 PRINT
80 PRINT"
{OR}  A - ENGINE WON'T TURN OVER"
90 PRINT"
{CYN}  B - ENGINE TURNS, BUT WON'T START
"
100 PRINT"
{PUR}  C - STARTS THEN STALLS STRAIGHT A
WAY"
110 PRINT"
{GRN}  D - CAR RUNS, BUT STALLS A LOT"
120 PRINT"
{BLU}  E - CAR RUNS, BUT IDLES ROUGHLY{W
HT}"
130 GET A$:IF A$<>"" THEN 130
140 GET A$
```

```
150 IF A$<"A" OR A$>"E" THEN 140
160 GOSUB 1560
170 ON (ASC(A$)-64) GOTO 200,580,1200,12
30,1350
180 END
190 REM ************************
200 REM WON'T TURN OVER
210 PRINT "
{OR}WE'LL START BY CHECKING THE BATTERY.
"
220 PRINT "TURN ON THE LIGHTS."
230 PRINT TAB(6);"ARE THEY DIM?"
240 GOSUB 1520
250 IF A$="N" THEN 350:REM BATTERY OK
260 PRINT "ARE BATTERY CABLES LOOSE OR C
ORRODED?"
270 GOSUB 1520
280 IF A$="Y" THEN PRINT "TIGHTEN AND CL
EAN"
290 GOSUB 1580
300 PRINT "IS FAN BELT LOOSE?"
310 GOSUB 1520
320 IF A$="Y" THEN PRINT TAB(8);"TIGHTEN
 THE FAN BELT"
330 GOSUB 1580
340 PRINT"JUMPER LEADS OR A PUSH SHOULD
BE ENOUGH TO START THE CAR":END
350 PRINT "IS THERE LOOSENESS OR CORROSI
ON AT THE"
360 PRINT "STARTER END OF THE BATTERY CA
BLE"
370 GOSUB 1520
380 IF A$="Y" THEN PRINT "TIGHTEN AND CL
EAN THE CONNECTIONS"
390 GOSUB 1580
```

```
400 PRINT "PUT A PIECE OF METAL ACROSS
THE"
410 PRINT "SOLENOID TERMINALS. DOES STAR
TER WORK?"
420 GOSUB 1520
430 IF A$="Y" THEN PRINT "THE IGNITION S
WITCH IS PROBABLY FAULTY"
440 GOSUB 1580
450 IF A$="Y" THEN PRINT TAB(4);"IT SHOU
LD BE REPLACED":END
460 PRINT "DOES STARTER CLICK?"
470 GOSUB 1520
480 IF A$="Y" THEN PRINT "THE STARTER MA
Y BE JAMMED":GOTO 520
490 PRINT "AS NOTHING HAPPENED, THE SOLE
NOID IS"
500 PRINT"PROBABLY FAULTY. A PUSH MAY ST
ART"
510 PRINTTAB(12);"YOUR CAR.":END
520 REM STARTER JAMMED
530 PRINT"TURN THE IGNITION OFF, AND PUT
 THE CAR"
540 PRINT"IN A HIGH GEAR. PUSH CAR FOR A
 FOOT OR"
550 PRINT "SO TO POP STARTER LOOSE.":END
560 RETURN
570 REM *************************
580 REM TURN OVER, WON'T START
590 PRINT"
{CYN}CHECK WIRES AT POINTS FOR DAMPNESS.
"
600 PRINT"IS THERE A CHANCE AREAS ARE DA
MP?"
610 GOSUB 1520
620 IF A$="Y" THEN PRINT "SPRAY WITH MOI
```

```
STURE REMOVAL SPRAY":GOSUB 158Ø
63Ø PRINT"IS THERE DUST VISIBLE ON THE I
NSULATING"
64Ø PRINT TAB(6);"PART OF THE COIL, OR O
N THE"
65Ø PRINT TAB(11);"DISTRIBUTOR CAP?"
66Ø GOSUB 152Ø
67Ø IF A$="Y" THEN PRINT "WIPE COIL SECT
ION AS WELL AS INSIDE"
68Ø IF A$="Y" THEN PRINT TAB(6);"AND OUT
SIDE OF CAP.":GOSUB 158Ø
69Ø PRINT "ENSURE ALL WIRES ARE TIGHT AN
D DRY       BEFORE CONTINUING"
7ØØ GOSUB 158Ø
71Ø PRINT "IF CAR STILL DOES NOT START,
IT IS TIME TO CHECK THE SPARK PLUGS:"
72Ø GOSUB 158Ø
73Ø PRINT "DOES SPARK FROM THE END OF TH
E PLUG WIREJUMP 3/8 INCH OR MORE?"
74Ø GOSUB 152Ø
75Ø IF A$="Y" THEN PRINT "THE PLUGS ARE
FAULTY":GOSUB 158Ø
76Ø IF A$="N" THEN 83Ø
77Ø PRINT "ARE PLUGS GREASY?"
78Ø GOSUB 152Ø
79Ø IF A$="Y" THEN PRINT "AN EMERGENCY R
EPAIR CANNOT BE MADE"
8ØØ IF A$="Y" THEN PRINT "WHILE PLUGS AR
E IN THAT STATE.":GOSUB 158Ø:GOTO 77Ø
81Ø IF A$="N" THEN PRINT "IF THIS IS AN
EMERGENCY, TRY CLOSING"
82Ø IF A$="N" THEN PRINT "THE GAP TO ABO
UT HALF NORMAL":GOSUB 158Ø:GOTO 91Ø
83Ø PRINT"IS THERE ANY SPARK AT ALL?"
84Ø GOSUB 152Ø
```

```
850 IF A$="Y" THEN 770
860 GOSUB 870:END
870 PRINT"CHECK ROTOR, COIL AND DISTRIBU
TOR CAP"
880 PRINT"FOR CRACKS. IF THERE AREN'T AN
Y THERE,"
890 PRINT"IT LOOKS AS IF THE POINTS OR C
ONDENSER  IS YOUR PROBLEM"
900 PRINT"A REPAIR MAY WELL BE NEEDED":R
ETURN
910 PRINT"DO YOU HAVE GAS IN THE TANK?"
920 GOSUB 1520
930 IF A$="N" THEN PRINT"FILL TANK AND T
RY AGAIN":GOSUB 1580
940 PRINT "ARE ALL FUEL AND VACUUM LINES
 SECURE?"
950 GOSUB 1520
960 IF A$="N" THEN PRINT"ATTEND TO THESE
 AND TRY AGAIN":GOSUB 1580
970 PRINT"    REMOVE THE AIR CLEANER FROM
 THE"
980 PRINT TAB(13);"CARBURETOR":GOSUB 158
0
990 PRINT"DOES IT LOOK DRY?"
1000 GOSUB 1520
1010 IF A$="N" THEN 1080
1020 PRINT"TURN THE ENGINE OVER A FEW TI
MES WITH"
1030 PRINT"YOUR HAND SEALING THE AIR INT
AKE":GOSUB 1580
1040 PRINT "IS YOUR HAND WET WITH GAS?"
1050 GOSUB 1520
1060 IF A$="N" THEN PRINT"UNSCREW GAS CA
P IN CASE AIR VENT IS PLUGGED"
1070 IF A$="N" THEN PRINT"THE FUEL PUMP
```

```
MAY NOT BE WORKING":GOSUB 900:END
1080 PRINT"HAVE YOU BEEN CRANKING THE ST
ARTER A"
1090 PRINT"LOT IN THE PAST FEW MINUTES?"
1100 GOSUB 1520
1110 IF A$="N" THEN 1150
1120 PRINT "WAIT FOR A MINUTE OR SO, THE
N HOLD THE"
1130 PRINT "GAS PEDAL STEADILY ON THE FL
OOR WITHOUT"
1140 PRINT"PUMPING IT. THIS SHOULD GET Y
OU GOING":END
1150 PRINT"THE ENGINE MAY WELL BE FLOODE
D AT THIS"
1160 PRINT"POINT AND THE FLOAT VALVE STU
CK OPEN.":GOSUB 1580
1170 PRINT"TAP THE SIDE OF THE CARBURETO
R"
1180 PRINT"THEN TRY STARTING PROCESSES A
GAIN"
1190 REM ***********************
1200 REM STARTS, THEN STALLS
1210 PRINT"
{PUR}THIS SUGGESTS A FAULTY BALLAST RESI
STOR WHICH SHOULD BE REPLACED":END
1220 REM ***********************
1230 REM RUNS, STALLS A LOT
1240 PRINT"
{GRN}THE PROBLEM IS EITHER CAUSED BY:"
1250 PRINT TAB(7);"SHORTING (OR LOOSE) W
IRES;"
1260 PRINT TAB(7);"A WEAK SPARK; OR"
1270 PRINT TAB(7);"A FAULT IN THE FUEL S
YSTEM"
1280 PRINT"CHECK FIRST FOR LOOSE OR SHOR
TING WIRES":GOSUB 1580
```

```
1290 PRINT "IF THEY ARE NOT OK, REPAIR.
IF THEY ARE IT COULD BE THE SPARK PLUGS"
1300 GOSUB 870
1310 PRINT"THERE IS A FINAL CHECK WE CAN
 TRY ON"
1320 PRINT"YOUR SPARK PLUGS":GOSUB 1580
1330 GOTO 1360
1340 REM ************************
1350 REM RUNS, ROUGH IDLE
1360 PRINT"
{BLU}IT COULD WELL BE THAT ONE OR MORE O
F"
1370 PRINT"YOUR  SPARK PLUGS ARE FAULTY.
":GOSUB 1580
1380 PRINT"DISCONNECT THEM ONE AT A TIME
. THE ONES"
1390 PRINT"WHICH DO NOT CAUSE THE ENGINE
 IDLE TO"
1400 PRINT"DROP ARE FAULTY. CHECK THESE
THEN"
1410 PRINT"RETURN TO THE SYSTEM.":GOSUB
1580
1420 PRINT"DID TEST SHOW ANY PLUGS WERE
FAULTY?"
1430 GOSUB 1520
1440 IF A$="Y" THEN PRINT"REPLACE ALL PL
UGS IF YOU CAN, OR JUST"
1450 IF A$="Y" THEN PRINT"THE ONES WHICH
 TESTED FAULTY":GOSUB 1580
1460 PRINT"THE MOST COMMON CAUSE OF A BA
D IDLE,"
1470 PRINT "ASSUMING THAT THE PLUGS ARE
OK, IS THAT"
1480 PRINT"YOUR GAS MIXTURE IS SET TOO R
ICH"
```

```
1490 PRINT"SO YOU SHOULD ADJUST THIS":EN
D
1500 PRINT"ADJUST THE IDLE SPEED-SCREW O
N THE       THROTTLE LINKAGE":END
1510 REM ************************
1520 PRINTTAB(16);"(Y - YES, N - NO)?"
1530 GET A$:IF A$<>"" THEN 1530
1540 GET A$
1550 IF A$<>"Y" AND A$<>"N" THEN 1540
1560 PRINT TAB(22);" >  OK   ";A$;" <"
1570 REM *****
1580 FOR T=1 TO 1000:NEXT T
1590 PRINT
1600 RETURN
```

MEDICI

```
10 REM MEDICI- PERSONAL CHECKUP
20 POKE 53280,0:POKE 53281,0:PRINT "
{CLR}{WHT}"
30 GOSUB 1250
40 PRINT "
{PUR}A - I AM BADLY OVERWEIGHT"
50 PRINT "B - I AM FAIRLY OVERWEIGHT"
60 PRINT "C - I AM SLIGHTLY OVERWEIGHT"
70 PRINT "D - MY WEIGHT IS ABOUT RIGHT"
80 PRINT "E - I AM THINNER THAN I SHOULD
 BE
{WHT}"
90 GOSUB 1170
100 WEIGHT=5*(ASC(A$)-68):IF A$="E" THEN
 WEIGHT=0
110 PRINT WEIGHT
120 GOSUB 1250
130 PRINT "I ENGAGE IN EXERCISE, THAT"
```

```
140 PRINT"RAISES MY HEARTBEAT TO 120 OR
MORE,"
150 PRINT "FOR AT LEAST THE FOLLOWING NU
MBER"
160 PRINT TAB(8);"HOURS A WEEK:"
170 PRINT
180 PRINT "
{CYN}A - LESS THAN A QUARTER"
190 PRINT "B - MORE THAN A QUARTER, UP T
O                THREE-QUARTERS"
200 PRINT "C - FROM THREE-QUARTERS OF AN
 HOUR            UP TO ONE AND A HALF"
210 PRINT "D - FROM ONE AND A HALF TO
                 TWO AND A HALF"
220 PRINT "E - MORE THAN TWO AND A HALF
HOURS
{WHT}"
230 GOSUB 1170
240 EXERCISE=5*(ASC(A$)-63)-5:IF A$="A"
THEN EXERCISE=0
250 PRINT EXERCISE
260 GOSUB 1250
270 PRINT "WHEN DRIVING:":PRINT
280 PRINT "
{BLU}A - I HARDLY EVER WEAR A SEAT BELT"
290 PRINT "B - I WEAR A SEAT BELT AROUND
 A QUARTER       OF THE TIME"
300 PRINT "C - I WEAR A SEAT BELT EVERY
SECOND           JOURNEY"
310 PRINT "D - I WEAR A SEAT BELT MOST,
BUT NOT          ALL TRIPS"
320 PRINT "E - I ALWAYS WEAR A SEAT BELT
{WHT}"
330 GOSUB 1170
340 SEATBELT=2*(ASC(A$)-65)
350 PRINT SEATBELT
```

```
360 GOSUB 1250
370 PRINT "I AM CONSCIOUS OF NUTRITION A
ND TRY      TO EAT HEALTHILY:"
380 PRINT
390 PRINT "
{GRN}A - ALL THE TIME"
400 PRINT "B - NEARLY ALL THE TIME"
410 PRINT "C - A FAIR PROPORTION OF THE
TIME"
420 PRINT "D - FROM TIME TO TIME"
430 PRINT "E - HARDLY AT ALL
{WHT}"
440 GOSUB 1170
450 DIET=-ASC(A$)+69
460 PRINT DIET
470 GOSUB 1250
480 PRINT "SMOKING (A CIGAR COUNTS AS A
CIGARETTE)"
490 PRINT
500 PRINT "
{OR}A - NOT AT ALL"
510 PRINT "B - LESS THAN 15 CIGARETTES A
 DAY"
520 PRINT "C - 15 TO 25 CIGARETTES A DAY
"
530 PRINT "D - 26 TO 42 CIGARETTES A DAY
"
540 PRINT "E - MORE THAN 42 CIGARETTES A
 DAY
{WHT}"
550 GOSUB 1170
560 SMOKING=-7*(ASC(A$)-65)
570 PRINT SMOKING
580 GOSUB 1250
```

```
590 PRINT "ALCOHOL- HOW MANY DRINKS (ON
AVERAGE)    DO YOU HAVE EACH DAY?"
600 PRINT
610 PRINT "
{LT RED}A - NONE"
620 PRINT "B - LESS THAN 3"
630 PRINT "C - 3 TO 6"
640 PRINT "D - 7 TO 9"
650 PRINT "E - MORE THAN 9
{WHT}"
660 GOSUB 1170
670 DRINK=-30
680 IF A$="A" THEN DRINK=0
690 IF A$="B" THEN DRINK=1
700 IF A$="C" THEN DRINK=DRINK/5
710 IF A$="D" THEN DRINK=DRINK/2
720 PRINT DRINK
730 GOSUB 1250
740 PRINT "IN GENERAL, HOW STRESSFUL WOU
LD YOU SAY";
750 PRINT "YOUR LIFE HAS BEEN IN THE PAS
T 6 MONTHS"
760 PRINT
770 PRINT "
{LT GRN}A - EXTREMELY STRESSFUL"
780 PRINT "B - FAIRLY STRESSFUL"
790 PRINT "C - SLIGHTLY STRESSFUL"
800 PRINT "D - NUETRAL"
810 PRINT "E - NOT STRESSFUL
{WHT}"
820 GOSUB 1170
830 SRESS=INT(2.5*(ASC(A$)-69))
840 PRINT SRESS
850 GOSUB 1300:PRINT "
{CLR}"
```

```
860 PRINT "PERSONAL ASSESEMENT FROM MEDIC
I:"
870 PRINT
880 PRINT TAB(8);"WEIGHT:"WEIGHT
890 PRINT TAB(6);"EXERCISE:"EXERCISE
900 PRINT TAB(4);"CAR SAFETY:"SEATBELT
910 PRINT TAB(5);"NUTRITION:"DIET
920 PRINT TAB(7);"SMOKING:"SMOKING
930 PRINT TAB(7);"ALCOHOL:"DRINK
940 PRINT TAB(8);"STRESS:"SRESS
950 GOSUB 1300
960 ANT=WEIGHT+EXERCISE+SEATBELT+DIET+SM
OKING+DRINK+SRESS
970 GOSUB 1300:PRINT
980 PRINT "   YOUR RAW RATING IS "ANT:PR
INT
990 PRINT "  ON A SCALE WHERE ZERO IS AV
ERAGE,"
1000 PRINT"THE LOWEST RATING IS BELOW -8
0, AND"
1010 PRINT "  THE HIGHEST IS OVER 30"
1020 GOSUB 1300:PRINT
1030 IF ANT<6 AND ANT>-6 THEN A$="AVERAG
E":L$="60 TO 66   65 TO 71"
1040 IF ANT<-5 AND ANT>-21 THEN A$="BELO
W AVERAGE":L$="60 TO 66   65 TO 71"
1050 IF ANT<-20 THEN A$="POOR":L$="60 OR
 LESS 65 OR LESS"
1060 IF ANT<-45 THEN A$="VERY POOR"
1070 IF ANT<-60 THEN A$="VERY, VERY POOR
"
1080 IF ANT>5 AND ANT<15 THEN A$="GOOD":
L$="74 TO 80   79 TO 85"
1090 IF ANT>14 THEN A$="EXTREMELY GOOD":
L$="81 PLUS    86 PLUS"
```

```
1100 PRINT "THIS INDICATES YOUR HEALTH S
TATUS          IS ";A$
1110 PRINT
1120 PRINT "LIFE EXPECTANCY:"
1130 PRINT TAB(3);"MALE          FEMALE"
1140 PRINT TAB(3);L$
1150 END
1160 REM *********************
1170 REM ACCEPT INPUT
1180 GET A$:IF A$<>"" THEN 1180
1190 GET A$
1200 IF A$<"A" OR A$>"E" THEN 1190
1210 PRINT:PRINT TAB(12);"OK     ";A$
1220 RETURN
1230 REM *********************
1240 REM DELAY/SPACE OUT
1250 FOR J=1 TO 1000:NEXT J
1260 PRINT "
{CLR}"
1270 PRINT:PRINT:PRINT:PRINT
1280 PRINT "WHICH OF THE FOLLOWING IS CL
OSEST          TO THE TRUTH (SELECT ONE):"
1290 PRINT
1300 FOR J=1 TO 400:NEXT J
1310 RETURN
```

FUZZY RITA

```
10 REM FUZZY RITA
20 GOSUB 1380:REM INITIALISE
30 GOSUB 1160:REM OUTPUT OPTIONS
40 GOSUB 1250:REM DISCRIMINATION OPTIONS
50 GOSUB 140:REM QUESTION USER
60 GOSUB 480:REM MAKE DECISION AND
                 UPDATE KNOWLEDGE BASE
```

```
65 GOSUB 2ØØØ:REM MAKE SOUND
7Ø PRINT"
{PUR}PRESS <RETURN> TO CONTINUE{WHT}";
8Ø IF X$<>"" THEN INPUT I$:GOTO 5Ø
9Ø PRINT"
{PUR}TRAINING"
1ØØ PRINT"OR ANY KEY THEN <RETURN> TO US
E RITA
{WHT}";
11Ø INPUT X$:GOTO 5Ø
12Ø END
13Ø REM ****************************
14Ø REM QUESTION USER
15Ø PRINT"
{CLR}"
16Ø PRINT"
{PUR}HERE ARE THE SUBJECTS I CAN
      DISCRIMINATE BETWEEN:"
17Ø PRINT
18Ø FOR J=1 TO TT
19Ø PRINT" >    ";A$(J)
2ØØ NEXT J:PRINT "
{WHT}"
21Ø GOSUB 15ØØ
22Ø IF X$="" THEN PRINT "
{PUR}THINK OF ONE, THEN PRESS <RETURN>{W
HT}"
23Ø IF X$<>"" THEN PRINT"
{PUR}I AM READY NOW TO DETERMINE WHICH O
NE   YOU HAVE{WHT}"
24Ø IF X$="" THEN INPUT X$
25Ø ADD=.5
26Ø FOR J=1 TO DQ
27Ø ADD=ADD+ADD
28Ø GOSUB 15ØØ
```

```
290 IF X$<>"" AND TT>2 THEN 390:REM
    CHECK IF QUESTION CAN BE JUMPED
300 PRINT"
{PUR} ENTER A NUMBER FROM"
310 PRINT"1 (TRUE) TO 0 (FALSE) ($ TO EN
D RUN)"
320 PRINT:PRINT E$(J);"
{WHT}"
330 INPUT H$:IF H$="$" THEN PRINT:PRINT
"
{PUR}THANK YOU{WHT}":PRINT:END
340 C(J)=VAL(H$)
350 C(J)=ADD*C(J)
360 NEXT J
370 RETURN
380 REM ****************************
390 REM CHECK IF QUESTION CAN BE
                JUMPED
400 JUMP=1
410 FOR  W=1 TO TT
420 IF ABS(B(W,J)-B(1,J))>.7 THEN JUMP=0
430 NEXT W
440 IF JUMP=0 THEN 300
450 C(JUMP)=B(W,J)
460 GOTO 360
470 REM ****************************
480 REM MAKE DECISION
490 FOR J=1 TO TT
500 D(J)=0:E(J)=0:F(J)=0
510 NEXT J
520 ADD=.5
530 FOR J=1 TO TT
540 ADD=ADD+ADD
550 FOR X=1 TO DQ
```

```
560 REM PLAY WITH VALUES IN NEXT THREE
        LINES FOR MOST EFFICIENT RESULTS
570 IF C(X)=B(J,X) THEN D(J)=D(J)+1
580 IF ABS(C(X)-B(J,X))<.6*ADD THEN E(J)
=E(J)+.4
590 IF ABS(C(X)-B(J,X))<1.2*ADD THEN F(J
)=F(J)+.1
600 NEXT X
610 NEXT J
620 A1=1:A2=1:A3=1
630 F1=1:F2=1:F3=1
640 FOR J=1 TO TT
650 IF D(J)>F1 THEN F1=D(J):A1=J
660 IF E(J)>F2 THEN F2=E(J):A2=J
670 IF F(J)>F3 THEN F3=F(J):A3=J
680 NEXT J
690 REM ** ANNOUNCE RESULT **
700 PRINT
710 CFLG=0
720 PRINT"
{PUR}THE MOST LIKELY RESULT IS ";A$(A1)
730 IF A2<>A1 THEN PRINT"THE NEXT MOST L
IKELY IS ";A$(A2):CFLAG=1
740 IF A3<>A2 AND A3<>A1 THEN PRINT"THE
NEXT MOST LIKELY IS ";A$(A3):CFLAG=2
750 PRINT
760 PRINT"IS THE MOST LIKELY RESULT CORR
ECT                    (Y OR N)
{WHT}";
770 INPUT F$:F$=RIGHT$(F$,1)
780 IF F$<>"Y" AND F$<>"N" THEN 770
790 IF F$<>"Y" AND X$<>"" THEN RETURN
800 IF F$="Y" THEN 980
810 IF TT=2 AND A1=1 THEN A1=2:GOTO 980
820 IF TT=2 THEN A1=1:GOTO 980
830 IF CFLG=0 THEN 890
```

```
840 PRINT"
{PUR}IS MY SECOND CHOICE CORRECT
          (Y OR N){WHT}";
850 INPUT F$:F$=RIGHT$(F$,1)
860 IF F$="N" THEN 890
870 IF CFLG=1 THEN A1=A2:GOTO 980
880 IF CFLG=2 THEN A1=A3:GOTO 980
890 GOSUB 1500
900 FOR J=1 TO TT
910 PRINTJ;"- ";A$(J)
920 NEXT J
930 PRINT
940 PRINT"
{PUR}WHICH IS THE CORRECT ONE{WHT}"
950 INPUT A1
960 IF A1<1 OR A1>TT THEN 950
970 REM     ** EDUCATING RITA **
           (UPDATE KNOWLEDGE BASE)
980 FOR J=1 TO DQ
990 IF B(A1,J)<>0 THEN B(A1,J)=(C(J)+5*B
(A1,J))/6
1000 IF B(A1,J)=0 THEN B(A1,J)=C(J)
1010 B(A1,J)=INT(10*B(A1,J))/10
1020 NEXT J
1030 PRINT
1040 IF U$="" THEN RETURN
1050 FOR J=1 TO TT
1060 PRINT:GOSUB 1500
1070 PRINT A$(J)
1080 PRINT
1090 FOR K=1 TO DQ
1100 PRINT E$(K);B(J,K)
1110 NEXT K
1120 NEXT J
1130 PRINT
```

```
1140 RETURN
1150 REM ****************************
1160 REM OUTPUT OPTIONS
1170 TT=0
1180 TT=TT+1
1190 GOSUB 1500
1200 PRINT"
{PUR}ENTER OUTPUT OPTION NUMBER"TT"
      (PRESS <RETURN> TO END){WHT}"
1210 INPUT A$(TT)
1220 IF A$(TT)="" OR TT=51 THEN TT=TT-1:
RETURN
1230 GOTO 1180
1240 REM ****************************
1250 REM DISCRIMINATION QUESTIONS
1260 PRINT"
{CLR}{PUR}"
1270 FOR J=1 TO TT
1280 PRINT A$(J)
1290 NEXT J
1300 DQ=0
1310 DQ=DQ+1
1320 GOSUB 1500
1330 PRINT"
{PUR}ENTER QUESTION NUMBER "DQ:PRINT"
 (PRESS <RETURN> TO END){WHT}"
1340 INPUT E$(DQ)
1350 IF E$(DQ)="" OR DQ=51 THEN DQ=DQ-1:
RETURN
1360 GOTO 1310
1370 REM ****************************
1380 REM INITIALISATION
1390 PRINT"
{CLR}":POKE 53280,0:POKE 53281,0
```

```
1400 REM REDUCE ARRAYS IN NEXT LINE IN
         ACCORDANCE WITH YOUR NEEDS
1410 DIM A$(50),B(50,50),C(50),D(50),E$(
50),F(50),G(50)
1420 X$=""
1430 PRINT"
{PUR}PRESS ANY KEY, THEN <RETURN> IF YOU
"
1440 PRINT"WANT TOSEE UPDATED KNOWLEDGE
BASE"
1450 PRINT"      AFTER EACH RUN; JUST PRE
SS"
1460 PRINT"        <RETURN> IF YOU DON'T
{WHT}"
1470 INPUT U$
1480 PRINT"
{CLR}"
1490 RETURN
1500 PRINT"
{PUR}-----------------------------------
----{WHT}"
1510 RETURN
2000 SID=54272
2010 FOR L1=0 TO 23
2020 POKE SID+L1,0
2030 NEXT L1
2040 POKE SID+24,7
2050 POKE SID+5,15
2060 POKE SID+6,55
2070 POKE SID+4,33
2080 FOR L1=1 TO 40 STEP 4
2090 POKE SID+1,L1
2100 FOR L2=1 TO 10
2110 NEXT L2,L1
2120 POKE SID+6,0
2130 RETURN
```

SCALING

```
10 REM SCALING
20 DIM X(50),Z(50)
30 PRINT"
{CLR}"
40 INPUT "HIGHEST VALUE";A
50 INPUT "LOWEST VALUE";B
60 A=A+.001
70 B=B+.001
80 C=(A-B)/50
90 X(0)=B
100 FOR J=1 TO 50
110 X(J)=X(J-1)+C
120 Z(J)=J/50
130 PRINT Z(J),X(J)
140 NEXT J
150 DFF=(X(2)-X(1))/2
160 COUNT=0
170 COUNT=COUNT+1
180 PRINT"ENTER VALUE";Q$
190 INPUT Q$
200 IF Q$="" THEN END
210 Q=VAL(Q$)
220 IF Q<B OR Q>A THEN 180
230 FOR J=1 TO 50
240 IF ABS(Q-X(J))<DFF THEN PRINTCOUNT;"
-";Z(J)
250 NEXT J
260 GOTO 170
```

HASTE

```
10 REM HASTE
20 DIM A$(255),B$(255):F=0
30 PRINT"
{CLR}":POKE 53280,0:POKE 53281,0
40 REM ************************
50 FLAG=0
60 D$="":INPUT"
{PUR}> ";D$:PRINT"{WHT}"
70 IF D$="" THEN END
80 IF LEFT$(D$,1)="?" THEN 200
90 E=0
100 E=E+1
110 IF MID$(D$,E,1)="*" THEN 140
120 IF E<LEN(D$)  THEN 100
130 PRINT"INVALID ENTRY":GOTO 50
140 IF  FLAG=3 THEN RETURN
150 F=F+1:IF F=256 THEN END
160 A$(F)=LEFT$(D$,E-1)
170 B$(F)=MID$(D$,E+1)
180 GOTO 50
190 REM ************************
200 REM INTERROGATE
210 FLAG=4:TRUE=0
220 IF RIGHT$(D$,1)="/" THEN FLAG=3
230 FOR J=1 TO LEN(D$)-5
240 IF MID$(D$,J,5)=" AND " THEN FLAG=15
:TRUE=J
250 NEXT J
260 IF FLAG=5 THEN 410
270 IF LEFT$(D$,3)="?/*" THEN FLAG=1
280 IF LEFT$(D$,4)="?/*/" THEN FLAG=2
290 IF FLAG=3 THEN GOSUB 90:F$=MID$(D$,2
,E-2)
300 IF FLAG=1 THEN F$=MID$(D$,4)
```

```
310 E=0
320 E=E+1
330 IF A$(E)="" AND FLAG=4 AND TRUE=0 TH
EN PRINT"FALSE"
340 IF A$(E)="" THEN 520
350 IF FLAG=4 AND "?"+A$(E)+"*"+B$(E)=D$
 THEN PRINT"TRUE":TRUE=1
360 IF FLAG=3 AND F$=A$(E) THEN PRINT B$
(E)
370 IF FLAG=2 THEN PRINT A$(E);"*";B$(E)
380 IF FLAG=1 AND F$=B$(E) THEN PRINT A$
(E)
390 IF E<255 THEN 320
400 REM ***********************
410 F$=MID$(D$,4,TRUE-4):G$=MID$(D$,TRUE
+7)
420 E=0
430 E=E+1
440 IF A$(E)="" THEN 520
450 IF B$(E)=F$ THEN 470
460 IF E<255 THEN 430
470 H=0
480 H=H+1
490 IF B$(H)="" THEN 460
500 IF B$(H)=G$ AND A$(E)=A$(H) THEN PRI
NT A$(E)
510 IF H<255 THEN 480
520 PRINT TAB(5);" > END OF ANSWER <  "
530 GOSUB 2000:GOTO 50
2000 SID=54272
2010 FOR L1=0 TO 23
2020 POKE SID+L1,0
2030 NEXT L1
2040 POKE SID+24,15
2050 POKE SID+5,15
2060 POKE SID+6,255
```

```
2070 POKE SID+4,17
2080 FOR L1=48 TO 220 STEP .7
2090 POKE SID+1,L1
2100 NEXT L1
2110 FOR L1=28 TO 200
2120 POKE SID+1,L1
2130 NEXT L1
2140 FOR L1=200 TO 28 STEP -1
2150 POKE SID+1,L1
2160 NEXT L1
2170 POKE SID+1,0
2180 RETURN
```

EASLE

```
10 REM EASLE
20 PRINT"
{CLR}":POKE 53280,0:POKE 53281,0
30 DIM X(40),Y(40),Z(40)
40 PRINT
50 A$="":INPUT ">
{WHT}";A$
60 IF A$="" THEN END
70 REM **************************
80 PRINT"
{PUR}";:FOR J=1 TO 40
90 X(J)=0:Y(J)=0:Z(J)=0
100 NEXT J
110 REM **************************
120 R=0:S=0:T=0:CFIRST=0:CSECND=0:EDGE=0
130 FOR J=1 TO LEN(A$)
140 B$=MID$(A$,J,1)
150 IF B$="(" THEN S=S+1:Z(S)=J:IF T=0 T
HEN CFIRST=J
```

```
160 IF B$=")" THEN T=T+1:Y(T)=J:IF CSECN
D<>0 AND EDGE=0 THEN EDGE=J-1
170 IF T=1 AND B$=")" THEN CSECND=J
180 IF B$=" "  THEN R=R+1:X(R)=J
190 NEXT J
200 IF S=T THEN 260:REM ( ) BALANCE
210 IF S<T THEN PRINT " -> MISSING ("
220 IF S>T THEN PRINT " -> MISSING )"
230 INPUT ">
{WHT}";B$:PRINT "{PUR}";
240 A$=A$+B$
250 GOTO 80
260 FLAG=0
270 IF LEFT$(A$,5)="CAR (" THEN FLAG=1
280 IF LEFT$(A$,5)="CDR (" THEN FLAG=2
290 IF LEFT$(A$,6)="CONS (" THEN FLAG=3
300 IF LEFT$(A$,6)="ATOM (" THEN FLAG=4
310 IF LEFT$(A$,4)="EQ (" THEN FLAG=5
320 IF LEFT$(A$,6)="NULL (" THEN FLAG=6
330 ON FLAG GOSUB 420,470,550,690,780,92
0
340 IF FLAG<>0 THEN GOSUB 360
350 GOTO 40
360 REM ** RETURN ANSWER **
370 GOSUB 4000:PRINT"   VALUE IS..."
380 IF B$<>"()" THEN PRINT"    ";B$
390 IF B$="()" THEN PRINT"   NIL"
400 RETURN
410 REM *************************
420 REM     ** CAR **
430 IF S=2 THEN B$=MID$(A$,Z(2)+1,X(2)-Z
(2))
440 IF S>2 THEN B$=MID$(A$,CFIRST,CSECND
-CFIRST+1)
450 RETURN
```

```
460 REM **************************
470 REM      ** CDR **
480 GOSUB 420
490 LB=LEN(B$)+7
500 B$="("+MID$(A$,LB,EDGE-1)
510 IF RIGHT$(B$,2)="))" THEN B$=LEFT$(B
$,LEN(B$)-1)
520 IF MID$(B$,2,1)=" " THEN B$="("+MID$
(B$,3)
530 RETURN
540 REM **************************
550 REM      ** CONS **
560 B$=MID$(A$,7,LEN(A$)-1)
570 J=0
580 IF LEFT$(B$,1)="(" THEN J=1
590 J=J+1
600 IF MID$(B$,J,1)="(" THEN 630
610 IF J<LEN(B$) THEN 590
620 B$="     > CONS ERROR <":RETURN
630 LB=LEN(B$)-1
640 B$="("+LEFT$(B$,J-1)+MID$(B$,J+1)
650 B$=LEFT$(B$,LB)
660 IF RIGHT$(B$,2)=" )" THEN B$=LEFT$(B
$,LEN(B$)-2)+")"
670 RETURN
680 REM **************************
690 REM      ** ATOM **
700 A$=MID$(A$,7,LEN(A$)-1)
710 J=0:B$="NIL"
720 J=J+1
730 IF MID$(A$,J,1)=" " OR MID$(A$,J,1)=
"(" THEN RETURN
740 IF J<LEN(A$) THEN 720
750 B$="T"
760 RETURN
770 REM **************************
```

```
780 REM        ** EQ **
790 A$=MID$(A$,5)
800 A$=LEFT$(A$,LEN(A$)-1)
810 J=Ø:B$="NIL"
820 J=J+1
830 IF MID$(A$,J,1)=")" THEN RETURN
840 IF MID$(A$,J,1)=" " THEN 870
850 IF J<LEN(A$) THEN 820
860 RETURN
870 C$=LEFT$(A$,J-1)
880 A$=MID$(A$,J+1)
890 IF C$=A$ THEN B$="T"
900 RETURN
910 REM **************************
920 REM       ** NULL **
930 B$="NIL"
940 IF A$="NULL ()" THEN PRINT"ILLEGAL -
 NULL NEEDS ARGUMENT":B$=""
950 IF A$="NULL (())" THEN B$="T"
960 RETURN
4000 SID=54272
4010 FOR L1=Ø TO 23
4020 POKE SID+L1,Ø
4030 NEXT L1
4040 POKE SID+24,15
4050 POKE SID+5,15
4060 POKE SID+6,255
4070 POKE SID+4,17
4080 FOR L1=48 TO 220 STEP 3
4090 POKE SID+1,L1
4100 NEXT L1
4110 FOR L1=28 TO 200 STEP 3
4130 NEXT L1
4140 FOR L1=200 TO 28 STEP -3
4150 POKE SID+1,L1
```

```
4160 NEXT L1
4170 POKE SID+1,0
4180 RETURN
```

PROLOG-A

```
1 REM ***********************************
2 REM USE "." INSTEAD OF ":" . FOR
3 REM EXAMPLE, "WHICH(X : SUM(1 1 2))"
4 REM BECOMES "WHICH(X . SUM(1 1 2))"
5 REM ***********************************
10 REM PROLOG-A (SIMPLE FRONT-END)
20 REM * ALL INPUT IN UPPER CASE *
30 GOTO 50
40 PRINT"
{BLU}NO (MORE) ANSWERS{WHT}":RETURN
50 GOSUB 3270:REM INITIALISE
60 REM ****************************
70 GOSUB 4000:PRINT
80 J$="":INPUT "
{PUR}&.{WHT}";J$
90 IF J$="" THEN END
100 PRINT "
{PUR}";:IF J$="LIST ALL" THEN GOSUB 1860
:GOTO 70
110 IF LEFT$(J$,5)="LIST " THEN J$=MID$(
J$,5)+" ":GOSUB 990:GOTO 70
120 IF RIGHT$(J$,1)<>")" THEN PRINT"1.":
INPUT M$:J$=J$+M$:GOTO 120
130 LJ=LEN(J$)
140 J$=LEFT$(J$,LJ-1)+" ":REM STRIP FINA
L ), REPLACE WITH SPACE
150 LJ=LEN(J$)
160 FLAG=0
```

```
170 IF LEFT$(J$,4)="ADD(" THEN J$=MID$(J
$,5):FLAG=1
180 RULEFLAG=0:PLUSFLAG=0:ARITHFLAG=0
190 FOR R=1 TO LEN(J$)
200 IF MID$(J$,R,4)=" IF " THEN RULEFLAG
=R:FLAG=6
210 IF MID$(J$,R,5)=" AND " THEN PLUSFLA
G=R
220 IF MID$(J$,R,4)="SUM(" THEN ARITHFLA
G=1
230 IF MID$(J$,R,6)="TIMES(" THEN ARITHF
LAG=2
240 IF MID$(J$,R,6)=" LESS " THEN ARITHF
LAG=3
250 IF MID$(J$,R,3)="INT" THEN ARITHFLAG
=4
260 NEXT R
270 IF LEFT$(J$,3)="IS(" THEN J$=MID$(J$
,4):FLAG=2
280 IF LEFT$(J$,10)="WHICH(X . " THEN J$
=MID$(J$,11):FLAG=3
290 IF LEFT$(J$,16)="WHICH((X Y) . X " T
HEN J$=MID$(J$,17):FLAG=4
300 IF FLAG=0 THEN PRINT"SYNTAX ERROR":G
OTO 70
310 LJ=LEN(J$)
320 REM NOW SEND TO RELEVANT SUBROUTINES
330 IF PLUSFLAG<>0 THEN GOSUB 1950:GOTO
70:REM ENCODE RULE CONTAINING AND
340 IF RULEFLAG<>0 AND FLAG<>5 THEN GOSU
B 1110:REM ENCODE RULE
350 IF ARITHFLAG<>0 THEN GOSUB 2430:GOTO
 70:REM ARITHMETIC
360 IF RIGHT$(J$,3)=" X " OR RIGHT$(J$,3
)=" Y " THEN J$=LEFT$(J$,LJ-2)+" "
```

```
370 LJ=LEN(J$)
380 IF FLAG=1 THEN GOSUB 440:REM ADD
390 IF FLAG=2 THEN GOSUB 520:REM IS
400 IF FLAG=3 THEN GOSUB 610:REM WHICH
410 IF FLAG=4 THEN GOSUB 830:REM WHICH2
420 GOTO 70
430 REM ****************************
440 REM              ADD
450 K=0
460 K=K+1
470 IF Z$(K)="" THEN Z$(K)=J$:RETURN
480 IF K<1000 THEN 460
490 PRINT"MEMORY FULL"
500 RETURN
510 REM ****************************
520 REM              IS
530 K=0
540 K=K+1
550 IF Z$(K)="" THEN 580
560 IF Z$(K)=J$ THEN PRINT"YES":GOTO 590
570 IF K<1000 THEN 540
580 PRINT"NO"
590 RETURN
600 REM ****************************
610 REM            WHICH
620 IF LEFT$(J$,1)="X" THEN 710
630 J$=LEFT$(J$,LJ-1)
640 K=0
650 K=K+1
660 IF Z$(K)="" THEN 690
670 IF J$=LEFT$(Z$(K),LEN(J$)) THEN PRIN
T RIGHT$(Z$(K),(LEN(Z$(K))-LEN(J$)))
680 IF K<1000 THEN 650
690 GOSUB 40
```

```
700 RETURN
710 REM * QUERY STARTS WITH X *
720 J$=MID$(J$,3,LEN(J$)-3)
730 LJ=LEN(J$)
740 K=0
750 K=K+1
760 IF Z$(K)="" THEN 800
770 Q$=MID$(Z$(K),LEN(Z$(K))-LJ,LJ)
780 IF Q$=J$ THEN PRINT LEFT$(Z$(K),LEN(
Z$(K))-(LJ+2))
790 IF K<1000 THEN 750
800 GOSUB 40
810 RETURN
820 REM ****************************
830 REM             WHICH2
840 J$=LEFT$(J$,LJ-2)
850 LJ=LEN(J$)
860 K=0
870 K=K+1
880 IF Z$(K)="" THEN 960
890 LFLAG=0
900 FOR L=1 TO (LEN(Z$(K))-LJ)
910 IF MID$(Z$(K),L,LJ)=J$ THEN LFLAG=L
920 NEXT L
930 IF LFLAG=0 THEN 950
940 PRINT LEFT$(Z$(K),LFLAG-2);MID$(Z$(K
),(LFLAG+LJ))
950 IF K<1000 THEN 870
960 GOSUB 40
970 RETURN
980 REM ****************************
990 REM             LIST
1000 K=0
1010 K=K+1
```

```
1020 IF Z$(K)="" THEN RETURN
1030 LFLAG=0
1040 FOR L=1 TO (LEN(Z$(K))-LEN(J$))
1050 IF MID$(Z$(K),L,LEN(J$))=J$ THEN LF
LAG=1
1060 NEXT L
1070 IF LFLAG=1 THEN PRINT Z$(K)
1080 IF K<1000 THEN 1010
1090 RETURN
1100 REM ****************************
1110 REM       FORM RULES
1120 R=RULEFLAG
1130 E$=LEFT$(J$,R):F$=MID$(J$,R+4)
1140 IF LEFT$(E$,1)<>"X" THEN PRINT"RULE
 ERROR":GOTO 70
1150 REM NEXT LINE DETECTS INPUTS LIKE
         X EATS Y IF X IS-A Y
1160 IF RIGHT$(F$,2)="Y " THEN 1390
1170 PRINT TAB(18);"COMPILING RULE"
1180 FOR T=1 TO 100
1190 R$(T)=""
1200 NEXT T
1210 E$=MID$(E$,3):F$=MID$(F$,3)
1220 K=0:RR=0
1230 K=K+1
1240 IF Z$(K)="" THEN 1300
1250 IF RIGHT$(Z$(K),LEN(F$))<>F$ THEN 1
370
1260 RR=RR+1
1270 R$(RR)=LEFT$(Z$(K),(LEN(Z$(K))-LEN(
F$)))+E$
1280 PRINT "> ";R$(RR)
1290 GOTO 1230
1300 IF RR=0 THEN RETURN
1310 RC=0
1320 RC=RC+1
```

```
1330 Z$(K)=R$(RC)
1340 IF K<1000 THEN K=K+1
1350 IF RC<RR THEN 1320
1360 RETURN
1370 IF K<1000 THEN 1230
1380 RETURN
1390 REM * RULE WITH 2 VARIABLES *
1400 FOR T=1 TO 100
1410 R$(T)=""
1420 NEXT T
1430 K=0:RR=0
1440 IF K=1000 THEN RETURN
1450 K=K+1
1460 IF Z$(K)="" THEN 1770
1470 REM SPLIT INTO THREE WORDS
1480 Q$=Z$(K)
1490 J=0
1500 J=J+1
1510 IF MID$(Q$,J,1)=CHR$(32) THEN 1540
1520 IF J<LEN(Q$) THEN 1500
1530 PRINT"RULE COMPILING ERROR":GOTO 70
1540 A$=LEFT$(Q$,J)
1550 Q$=MID$(Q$,J+1)
1560 J=0
1570 J=J+1
1580 IF MID$(Q$,J,1)=CHR$(32) THEN 1610
1590 IF J<LEN(Q$) THEN 1570
1600 PRINT"RULE COMPILING ERROR":GOTO 70
1610 B$=LEFT$(Q$,J)
1620 Q$=MID$(Q$,J+1)
1630 J=0
1640 J=J+1
1650 IF MID$(Q$,J,1)=CHR$(32) THEN 1680
1660 IF J<LEN(Q$) THEN 1640
1670 PRINT"RULE COMPILING ERROR":GOTO 70
```

```
1680 PRINT TAB(18);"COMPILING RULE"
1690 C$=LEFT$(Q$,J)
1700 M$=MID$(F$,3,LEN(B$))
1710 IF B$<>M$ THEN 1440
1720 RR=RR+1
1730 N$=MID$(E$,3,LEN(E$)-4)
1740 R$(RR)=A$+N$+C$
1750 PRINT"> ";R$(RR)
1760 GOTO 1440
1770 IF RR=0 THEN RETURN
1780 M=0
1790 M=M+1
1800 IF M>RR THEN RETURN
1810 Z$(K)=R$(M)
1820 IF K=1000 THEN PRINT "OUT OF MEMORY
":GOTO 70
1830 K=K+1
1840 GOTO 1790
1850 REM ****************************
1860 REM           LIST ALL
1870 PRINT
1880 K=0
1890 K=K+1
1900 IF Z$(K)="" THEN RETURN
1910 PRINTZ$(K)
1920 IF K<1000 THEN 1890
1930 RETURN
1940 REM ****************************
1950 REM FORM RULES WITH 'AND' OF THE
           FOLLOWING TYPE:
1960 REM (X EATS Y IF X IS-A BIRD AND
           Y COMES IN BOXES)
1970 REM X STATEMENT MUST BE IN LIST
PRECEDING Y FOR ALL EXAMPLES TO BE CODED
1980 REM SPLIT INTO SECTIONS
1990 J$=MID$(J$,2):REM STRIP "X"
```

```
2000 PRINT TAB(20);"COMPILING RULE"
2010 J=1
2020 J=J+1
2030 IF MID$(J$,J,1)=CHR$(32) THEN 2060
2040 IF J<LEN(J$) THEN 2020
2050 PRINT"RULE COMPILING ERROR":RETURN
2060 A$=LEFT$(J$,J):REM RELATIONSHIP 1
2070 J$=MID$(J$,J+7):REM STRIP
     TO START OF SECOND RELATIONSHIP
2080 J=1:COUNT=0
2090 J=J+1
2100 IF MID$(J$,J,1)=CHR$(32) THEN COUNT
=COUNT+1
2110 IF COUNT=2 THEN 2140
2120 IF J<LEN(J$) THEN 2090
2130 PRINT"RULE COMPILING ERROR":RETURN
2140 B$=LEFT$(J$,J):REM  STATEMENT 1
2150 C$=MID$(J$,J+6):REM STATEMENT 2
2160 IF C$=CHR$(32) THEN PRINT"RULE COMP
ILING ERROR":RETURN
2170 REM NOW GO THROUGH DATABASE
2180 FOR T=1 TO 200
2190 R$(T)=""
2200 NEXT T
2210 R1=0:R2=99
2220 K=0
2230 K=K+1
2240 IF Z$(K)="" THEN 2310
2250 IF R1=99 OR R2=200 THEN PRINT"MEMOR
Y SHORTAGE":GOTO 2310
2260 LB=LEN(B$)
2270 IF RIGHT$(Z$(K),LB)=B$ THEN R1=R1+1
:R$(R1)=LEFT$(Z$(K),LEN(Z$(K))-LB)
2280 LC=LEN(C$)
2290 IF RIGHT$(Z$(K),LC)=C$ THEN R2=R2+1
:R$(R2)=LEFT$(Z$(K),LEN(Z$(K))-LC)
```

```
2300 IF K<1000 THEN 2230
2310 IF R$(100)="" THEN PRINT"STATEMENT
OF INPUT NOT IN DATABASE":RETURN
2320 REM NOW ENCODE RULES
2330 R1=0:R2=99
2340 R2=R2+1
2350 R1=R1+1
2360 IF R$(R1)>CHR$(32) AND R$(R2)>CHR$(
32) THEN Z$(K)=R$(R1)+A$+R$(R2)+" "
2370 PRINT"> ";Z$(K)
2380 K=K+1
2390 IF R$(R2+1)<>"" THEN 2340
2400 IF R$(R1+1)<>"" THEN 2350
2410 RETURN
2420 REM ****************************
2430 REM          ARITHMETIC
2440 LJ=LEN(J$)
2450 IF ARITHFLAG<3 THEN GOSUB 2490
2460 IF ARITHFLAG=3 THEN GOSUB 2890
2470 IF ARITHFLAG=4 THEN GOSUB 3080
2480 RETURN
2490 REM ******** SUM TIMES *********
2500 J$=MID$(J$,5,LJ-5)
2510 IF LEFT$(J$,2)="S(" THEN J$=MID$(J$
,3)
2520 LJ=LEN(J$)
2530 K=0
2540 K=K+1
2550 IF MID$(J$,K,1)=CHR$(32) THEN A$=LE
FT$(J$,K-1):J$=MID$(J$,K+1):GOTO 2580
2560 IF K<LJ THEN 2540
2570 PRINTTAB(12);"ARITHMETIC ERROR":RET
URN
2580 LJ=LEN(J$)
2590 K=0
```

```
2600 K=K+1
2610 IF MID$(J$,K,1)=CHR$(32) THEN B$=LE
FT$(J$,K-1):J$=MID$(J$,K+1):GOTO 2640
2620 IF K<LJ THEN 2600
2630 PRINTTAB(12);"ARITHMETIC ERROR":RET
URN
2640 LJ=LEN(J$)
2650 K=0
2660 K=K+1
2670 IF MID$(J$,K,1)=CHR$(41) THEN C$=LE
FT$(J$,K-1):GOTO 2700
2680 IF K<LJ THEN 2660
2690 PRINTTAB(12);"ERROR (TOO MANY VARIA
BLES)":RETURN
2700 AN=0:BN=0:CN=0
2710 IF ASC(A$)>58 THEN AN=1
2720 IF ASC(B$)>58 THEN BN=2
2730 IF ASC(C$)>58 THEN CN=4
2740 GUIDE=AN+BN+CN:IF GUIDE=3 OR GUIDE=
5 OR GUIDE=6 THEN 2690
2750 IF ARITHFLAG=2 THEN 2820:REM TIMES
2760 IF GUIDE>0 THEN 2790
2770 IF VAL(A$)+VAL(B$)=VAL(C$) THEN PRI
NT"YES":RETURN
2780 PRINT"NO":RETURN
2790 IF GUIDE=1 THEN PRINTVAL(C$)-VAL(B$
):GOSUB 40:RETURN
2800 IF GUIDE=2 THEN PRINTVAL(C$)-VAL(A$
):GOSUB 40:RETURN
2810 PRINT VAL(A$)+VAL(B$):GOSUB 40:RETU
RN
2820 REM ** TIMES **
2830 IF GUIDE>0 THEN 2860
2840 IF VAL(A$)*VAL(B$)=VAL(C$) THEN PRI
NT"YES":RETURN
2850 PRINT"NO":RETURN
```

```
2860 IF GUIDE=1 THEN PRINTVAL(C$)/VAL(B$
):GOSUB 40:RETURN
2870 IF GUIDE=2 THEN PRINTVAL(C$)/VAL(A$
):GOSUB 40:RETURN
2880 PRINT VAL(A$)*VAL(B$):GOSUB 40:RETU
RN
2890 REM ********* LESS *************
2900 NF=0
2910 IF ASC(J$)<58 THEN NF=1:REM NUMBERS
2920 COUNT=0
2930 K=0
2940 K=K+1
2950 IF MID$(J$,K,1)=CHR$(32) THEN COUNT
=COUNT+1
2960 IF COUNT=2 THEN 3000
2970 IF K<LEN(J$) THEN 2940
2980 PRINT TAB(20)"COMPARISON ERROR"
2990 RETURN
3000 B$=MID$(J$,K+1)
3010 A$=LEFT$(J$,K-6)
3020 IF NF=1 THEN 3050
3030 IF A$<B$ THEN PRINT "YES":RETURN
3040 PRINT"NO":RETURN
3050 REM * NUMBERS *
3060 IF VAL(A$)<VAL(B$) THEN PRINT"YES":
RETURN
3070 PRINT"NO":RETURN
3080 REM ********** INT *************
3090 IF RIGHT$(J$,2)="X " THEN 3190
3100 K=0
3110 K=K+1
3120 IF MID$(J$,K,1)=CHR$(32) THEN 3160
3130 IF K<LEN(J$) THEN 3110
3140 PRINT TAB(20);"ARITHMETIC ERROR"
3150 RETURN
```

```
3160 A=VAL(LEFT$(J$,K-1))
3170 IF INT(A)=A THEN PRINT"YES":RETURN
3180 PRINT"NO":RETURN
3190 K=0
3200 K=K+1
3210 IF MID$(J$,K,1)=CHR$(32) THEN 3240
3220 IF K<LEN(J$) THEN 3200
3230 PRINT TAB(20);"ARITHMETIC ERROR":RE
TURN
3240 PRINT INT(VAL(LEFT$(J$,K-1)))
3250 RETURN
3260 REM *****************************
3270 REM            INITIALISE
3280 PRINT "
{CLR}":POKE 53280,0:POKE 53281,0
3290 DIMZ$(1000),R$(200)
3300 RETURN
4000 SID=54272
4010 FOR L1=0 TO 23
4020 POKE SID+L1,0
4030 NEXT L1
4040 POKE SID+24,15
4050 POKE SID+5,15
4060 POKE SID+6,255
4070 POKE SID+4,17
4080 FOR L1=48 TO 220 STEP 3
4090 POKE SID+1,L1
4100 NEXT L1
4110 FOR L1=28 TO 200 STEP 3
4130 NEXT L1
4140 FOR L1=200 TO 28 STEP -3
4150 POKE SID+1,L1
4160 NEXT L1
4170 POKE SID+1,0
4180 RETURN
```

SSLISP

```
10 REM S.S. LISP
20 GOTO 3450
30 REM ***************************
40 A$=MID$(A$,E):A$=LEFT$(A$,LEN(A$)-1)
50 RETURN
60 REM ***************************
70 PRINT "
{PUR}"
80 NN=0
90 A$="":INPUT":
{WHT}";A$
100 IF A$="" THEN END:REM JUST PRESS
                    <RETURN> TO END RUN
110 REM ***************************
120 PRINT "
{PUR}";:FOR J=1 TO 12
130 X(J)=0:Y(J)=0:Z(J)=0
140 NEXT J
150 REM ***************************
160 R=0:S=0:T=0:CFIRST=0:CSECND=0:EDGE=0
170 FOR J=1 TO LEN(A$)
180 B$=MID$(A$,J,1)
190 IF B$="(" THEN S=S+1:Z(S)=J:IF T=0 T
HEN CFIRST=J
200 IF B$=")" THEN T=T+1:Y(T)=J:IF CSECN
D<>0 AND EDGE=0 THEN EDGE=J-1
210 IF T=1 AND B$=")" THEN CSECND=J
220 IF B$=" "  THEN R=R+1:X(R)=J
230 NEXT J
240 IF S=T THEN 300:REM ( ) BALANCE
250 IF S<T THEN PRINT " -> MISSING ("
260 IF S>T THEN PRINT " -> MISSING )"
270 INPUT "+ ";B$
280 A$=A$+B$
```

```
290 GOTO 120
300 IF NWDS=0 OR NN=1 THEN 370
310 M$=LEFT$(A$,X(1)-1)
320 FOR J=1 TO NWDS
330 IF M$=N$(J) THEN A$=O$(J)+MID$(A$,LE
N(N$(J))+1)
340 NEXT J
350 NN=1
360 GOTO 120
370 FLAG=0:B$="NIL"
380 IF LEFT$(A$,5)="CAR (" THEN FLAG=1
390 IF LEFT$(A$,5)="CDR (" THEN FLAG=2
400 IF LEFT$(A$,6)="CONS (" THEN FLAG=3
410 IF LEFT$(A$,6)="ATOM (" THEN FLAG=4
420 IF LEFT$(A$,4)="EQ (" THEN FLAG=5
430 IF LEFT$(A$,6)="NULL (" THEN FLAG=6
440 IF LEFT$(A$,8)="MEMBER (" THEN FLAG=
7
450 IF LEFT$(A$,6)="MEMQ (" THEN FLAG=8
460 IF LEFT$(A$,8)="APPEND (" THEN FLAG=
9
470 IF LEFT$(A$,9)="REVERSE (" THEN FLAG
=10
480 IF LEFT$(A$,7)="EQUAL (" THEN FLAG=1
1
490 IF LEFT$(A$,6)="LIST (" THEN FLAG=12
500 IF LEFT$(A$,8)="DEFINE (" THEN FLAG=
13
510 IF LEFT$(A$,6)="ADD1 (" THEN FLAG=14
520 IF LEFT$(A$,6)="SUB1 (" THEN FLAG=15
530 IF LEFT$(A$,7)="ZEROP (" THEN FLAG=1
6
540 IF LEFT$(A$,12)="DIFFERENCE (" THEN
FLAG=17
```

```
550 IF LEFT$(A$,6)="EXPT (" THEN FLAG=18
560 IF LEFT$(A$,5)="MAX (" THEN FLAG=19
570 IF LEFT$(A$,5)="MIN (" THEN FLAG=20
580 IF LEFT$(A$,6)="PLUS (" THEN FLAG=21
590 IF LEFT$(A$,7)="MINUS (" THEN FLAG=2
2
600 IF LEFT$(A$,10)="QUOTIENT (" THEN FL
AG=23
610 IF LEFT$(A$,7)="RECIP (" THEN FLAG=2
4
620 IF LEFT$(A$,11)="REMAINDER (" THEN F
LAG=25
630 IF LEFT$(A$,7)="TIMES (" THEN FLAG=2
6
640 IF LEFT$(A$,10)="GREATERP (" THEN FL
AG=27
650 IF LEFT$(A$,7)="LESSP (" THEN FLAG=2
8
660 IF LEFT$(A$,8)="MINUSP (" THEN FLAG=
29
670 IF LEFT$(A$,9)="NUMBERP (" THEN FLAG
=30
680 IF LEFT$(A$,6)="ONEP (" THEN FLAG=31
690 IF FLAG>13 THEN 720
700 ON FLAG GOSUB 840,890,970,1110,1200,
1330,1380,1510,1700,1760,2000,2130,3300
710 GOTO 760
720 IF FLAG>24 THEN 750
730 ON (FLAG-13) GOSUB 2180,2230,2280,23
50,2560,2600,2790,2820,2870,2920,2960
740 GOTO 760
750 ON (FLAG-24) GOSUB 3030,3070,3120,31
60,3200,3250,2280
760 IF FLAG<>0 THEN GOSUB 780
770 GOTO 70
```

```
78Ø REM ** RETURN ANSWER **
79Ø GOSUB 4ØØØ:PRINT"   VALUE IS..."
8ØØ IF B$<>"()" THEN PRINT"    ";B$
81Ø IF B$="()" THEN PRINT"   NIL"
82Ø RETURN
83Ø REM *************************
84Ø REM       ** CAR **
85Ø IF S=2 THEN B$=MID$(A$,Z(2)+1,X(2)-Z
(2))
86Ø IF S>2 THEN B$=MID$(A$,CFIRST,CSECND
-CFIRST+1)
87Ø RETURN
88Ø REM *************************
89Ø REM       ** CDR **
9ØØ GOSUB 84Ø
91Ø LB=LEN(B$)+7
92Ø B$="("+MID$(A$,LB,EDGE-1)
93Ø IF RIGHT$(B$,2)="))" THEN B$=LEFT$(B
$,LEN(B$)-1)
94Ø IF MID$(B$,2,1)=" " THEN B$="("+MID$
(B$,3)
95Ø RETURN
96Ø REM *************************
97Ø REM       ** CONS **
98Ø B$=MID$(A$,7,LEN(A$)-1)
99Ø J=Ø
1ØØØ IF LEFT$(B$,1)="(" THEN J=1
1Ø1Ø J=J+1
1Ø2Ø IF MID$(B$,J,1)="(" THEN 1Ø5Ø
1Ø3Ø IF J<LEN(B$) THEN 1Ø1Ø
1Ø4Ø B$="     > CONS ERROR <":RETURN
1Ø5Ø LB=LEN(B$)-1
1Ø6Ø B$="("+LEFT$(B$,J-1)+MID$(B$,J+1)
1Ø7Ø B$=LEFT$(B$,LB)
1Ø8Ø IF RIGHT$(B$,2)=" )" THEN B$=LEFT$(
B$,LEN(B$)-2)+")"
```

```
1090 RETURN
1100 REM ***************************
1110 REM      ** ATOM **
1120 A$=MID$(A$,7,LEN(A$)-1)
1130 J=0:B$="NIL"
1140 J=J+1
1150 IF MID$(A$,J,1)=" " OR MID$(A$,J,1)
="(" THEN RETURN
1160 IF J<LEN(A$) THEN 1140
1170 B$="T"
1180 RETURN
1190 REM ***************************
1200 REM      ** EQ **
1210 E=5:GOSUB 40
1220 J=0
1230 J=J+1
1240 IF MID$(A$,J,1)=")" THEN RETURN
1250 IF MID$(A$,J,1)=" " THEN 1280
1260 IF J<LEN(A$) THEN 1230
1270 RETURN
1280 C$=LEFT$(A$,J-1)
1290 A$=MID$(A$,J+1)
1300 IF C$=A$ THEN B$="T"
1310 RETURN
1320 REM ***************************
1330 REM      ** NULL **
1340 IF A$="NULL ()" THEN B$=" * ILLEGAL
 - NULL NEEDS ARGUMENT *"
1350 XX$="NULL (())":IF A$=XX$ THEN B$="
T"
1360 RETURN
1370 REM ***************************
1380 REM      ** MEMBER **
1390 C$=MID$(A$,9)
1400 J=1
```

```
1410 J=J+1
1420 IF MID$(C$,J,1)=")" OR MID$(C$,J,1)
="(" THEN D$=LEFT$(C$,J):GOTO 1450
1430 IF J<LEN(C$) THEN 1410
1440 RETURN
1450 J=LEN(D$)
1460 J=J+1
1470 IF MID$(C$,J,LEN(D$))=D$ THEN C$=LE
FT$(C$,LEN(C$)-1):GOTO 1630
1480 IF J<LEN(C$) THEN 1460
1490 RETURN
1500 REM *************************
1510 REM       ** MEMQ **
1520 C$=MID$(A$,7)
1530 J=0
1540 J=J+1
1550 XX$=" ":IF MID$(C$,J,1)=XX$ THEN 15
80
1560 IF J<LEN(A$) THEN 1540
1570 RETURN
1580 D$=LEFT$(C$,J)
1590 C$=MID$(C$,J+2)
1600 C$=LEFT$(C$,LEN(C$)-2)+" "
1610 J=0
1620 J=J+1
1630 IF MID$(C$,J,LEN(D$))=D$ THEN B$="(
"+MID$(C$,J):GOTO 1660
1640 IF J<LEN(C$) THEN 1620
1650 RETURN
1660 B$=LEFT$(B$,LEN(B$)-1)+")"
1670 IF RIGHT$(B$,3)=")))" THEN B$=LEFT$
(B$,LEN(B$)-1):GOTO 1670
1680 RETURN
1690 REM **************************
```

```
1700 REM      ** APPEND **
1710 B$=MID$(A$,9)
1720 B$=LEFT$(B$,Y(1)-9)+" "+MID$(B$,Z(3
)-7)
1730 B$=LEFT$(B$,LEN(B$)-1)
1740 RETURN
1750 REM ***************************
1760 REM      ** REVERSE **
1770 B$=""
1780 A$=MID$(A$,11):A$=LEFT$(A$,LEN(A$)-
2)
1790 CT=1
1800 J=0
1810 J=J+1:IF J>LEN(A$) THEN 1920
1820 XX$=MID$(A$,J,1):IF XX$=" " THEN GO
TO 1850
1830 IF XX$="(" THEN 1860
1840 GOTO 1810
1850 CT=CT+1:G$(CT)=LEFT$(A$,J-1):A$=MID
$(A$,J+1):GOTO 1800
1860 J=J+1:IF MID$(A$,J,2)="))" THEN 198
0
1870 IF MID$(A$,J,1)=")" THEN 1910
1880 IF J=LEN(A$) THEN 1900
1890 GOTO 1860
1900 CT=CT+1:G$(CT)=A$+")":GOTO 1930
1910 CT=CT+1:G$(CT)=LEFT$(A$,J):A$=MID$(
A$,J+1):GOTO 1800
1920 CT=CT+1:G$(CT)=A$
1930 FOR M=CT TO 1 STEP -1
1940 B$=B$+G$(M):IF M>1 THEN B$=B$+" "
1950 NEXT M
1960 B$="("+B$+")"
1970 RETURN
```

```
198Ø CT=CT+1:G$(CT)=LEFT$(A$,J+1):A$=MID
$(A$,J+2):GOTO 18ØØ
199Ø REM *************************
2ØØØ REM       ** EQUAL **
2Ø1Ø E=8:GOSUB 4Ø
2Ø2Ø M=ASC(A$):IF M>47 AND M<58 THEN 237
Ø
2Ø3Ø J=Ø
2Ø4Ø J=J+1
2Ø5Ø IF MID$(A$,J,2)=") " THEN J=J+1:GOT
O 128Ø
2Ø6Ø IF MID$(A$,J,3)=")))" THEN 21ØØ
2Ø7Ø IF MID$(A$,J,2)="))" THEN 211Ø
2Ø8Ø IF J<LEN(A$) THEN 2Ø4Ø
2Ø9Ø RETURN
21ØØ C$=LEFT$(A$,J+2):A$=MID$(A$,J+4):GO
TO 13ØØ
211Ø C$=LEFT$(A$,J+1):A$=MID$(A$,J+3):GO
TO 13ØØ
212Ø REM *************************
213Ø REM       ** LIST **
214Ø E=7:GOSUB 4Ø
215Ø B$="("+A$+")"
216Ø RETURN
217Ø REM *************************
218Ø REM       ** ADD1 **
219Ø E=7:GOSUB 4Ø
22ØØ B$=STR$(VAL(A$)+1)
221Ø RETURN
222Ø REM *************************
223Ø REM       ** SUB1 **
224Ø E=7:GOSUB 4Ø
225Ø B$=STR$(VAL(A$)-1)
226Ø RETURN
227Ø REM *************************
228Ø REM       ** ZEROP **
```

```
2290 IF FLAG=16 THEN E=8
2300 IF FLAG=31 THEN E=7
2310 GOSUB 40
2320 IF (A$="0" AND FLAG=16) OR (A$="1"
AND FLAG=31) THEN B$="T"
2330 RETURN
2340 REM ************************
2350 REM     ** TWO ARGUMENTS **
2360 E=13:GOSUB 40
2370 J=0
2380 J=J+1
2390 IF MID$(A$,J,1)=" " THEN 2420
2400 IF J<LEN(A$) THEN 2380
2410 B$=" * ERROR - ONLY ONE ARGUMENT *"
:RETURN
2420 P=VAL(LEFT$(A$,J-1))
2430 Q=VAL(MID$(A$,J+1))
2440 IF FLAG=17 THEN B$=STR$(P-Q):RETURN
2450 IF FLAG=23 OR FLAG=25 THEN B=P/Q
2460 IF FLAG=25 THEN B=INT(.5+Q*(B-INT(B
))*1000)/1000
2470 IF FLAG=18 THEN B=P^Q
2480 IF FLAG=11 AND P=Q THEN B$="T"
2490 IF FLAG=27 AND P>Q THEN B$="T"
2500 IF FLAG=28 AND P<Q THEN B$="T"
2510 IF FLAG=32 THEN B=P-Q
2520 IF FLAG=11 OR FLAG>26 THEN RETURN
2530 B$=STR$(B)
2540 RETURN
2550 REM ************************
2560 REM        ** EXPT **
2570 E=7:GOSUB 40
2580 GOTO 2370
2590 REM ************************
2600 REM ** MAX (MIN PLUS TIMES) *
```

```
2610 F$=LEFT$(A$,3):A$=MID$(A$,6)
2620 CT=0:FLAG=0
2630 IF F$="TIMES" THEN CT=1
2640 J=0
2650 J=J+1
2660 IF MID$(A$,J,1)=" " THEN 2690
2670 IF J<LEN(A$) THEN 2650
2680 IF J=LEN(A$) THEN FLAG=1
2690 P=VAL(LEFT$(A$,J-1)):IF FLAG=0 THEN
 A$=MID$(A$,J+1)
2700 IF F$<>"PLUS" AND CT=0 THEN CT=P
2710 IF F$="MAX" AND P>CT THEN CT=P
2720 IF F$="MIN" AND P<CT THEN CT=P
2730 IF F$="PLUS" THEN CT=CT+P
2740 IF F$="TIMES" THEN CT=CT*P
2750 IF FLAG=0 THEN 2640
2760 B$=STR$(CT)
2770 RETURN
2780 REM **************************
2790 REM        ** MIN **
2800 GOTO 2610
2810 REM **************************
2820 REM        ** PLUS **
2830 F$="PLUS"
2840 A$=MID$(A$,7)
2850 GOTO 2620
2860 REM **************************
2870 REM        ** MINUS **
2880 E=8:GOSUB 40
2890 B$=STR$(-VAL(A$))
2900 RETURN
2910 REM **************************
2920 REM     ** QUOTIENT **
2930 E=11:GOSUB 40
2940 GOTO 2370
2950 REM **************************
```

```
2960 REM        ** RECIP **
2970 E=8:GOSUB 40
2980 IF B$="0" THEN B$=" DIVISION BY ZER
O ILLEGAL":RETURN
2990 B=1/(VAL(A$))
3000 B$=STR$(B)
3010 RETURN
3020 REM *************************
3030 REM     ** REMAINDER **
3040 E=12:GOSUB 40
3050 GOTO 2370
3060 REM *************************
3070 REM        ** TIMES **
3080 F$="TIMES"
3090 A$=MID$(A$,8)
3100 GOTO 2620
3110 REM *************************
3120 REM     ** GREATERP **
3130 E=11:GOSUB 40
3140 GOTO 2370
3150 REM *************************
3160 REM        ** LESSP **
3170 E=8:GOSUB 40
3180 GOTO 2370
3190 REM *************************
3200 REM       ** MINUSP **
3210 E=9:GOSUB 40
3220 IF VAL(A$)<0 THEN B$="T"
3230 RETURN
3240 REM *************************
3250 REM       ** NUMBERP **
3260 A$=MID$(A$,10)
3270 IF ASC(A$)>44 AND ASC(A$)<58 THEN B
$="T"
3280 RETURN
```

```
3290 REM ***************************
3300 REM       ** DEFINE **
3310 A$=MID$(A$,9)
3320 F$=LEFT$(A$,X(2)-9)
3330 G$=MID$(A$,X(4)-6)
3340 J=0
3350 J=J+1
3360 IF MID$(G$,J,1)=" " THEN 3390
3370 IF J<LEN(G$) THEN 3350
3380 B$=" DEFINE ERROR":RETURN
3390 G$=LEFT$(G$,J-1)
3400 NWDS=NWDS+1
3410 O$(NWDS)=G$:N$(NWDS)=F$
3420 B$=F$
3430 RETURN
3440 REM ***************************
3450 REM INITIALISE
3460 PRINT"
{CLR}":POKE 53280,0:POKE 53281,0
3470 DIM G$(20),O$(20),N$(20),X(12),Y(12
),Z(12)
3480 NWDS=0:REM COUNT OF USER-DEFINED
                 'NEW WORDS'
3490 GOTO 70
4000 SID=54272
4010 FOR L1=0 TO 23
4020 POKE SID+L1,0
4030 NEXT L1
4040 POKE SID+24,15
4050 POKE SID+5,15
4060 POKE SID+6,255
4070 POKE SID+4,17
4080 FOR L1=48 TO 220 STEP 3
4090 POKE SID+1,L1
```

```
4100 NEXT L1
4110 FOR L1=28 TO 200 STEP 3
4130 NEXT L1
4140 FOR L1=200 TO 28 STEP -3
4150 POKE SID+1,L1
4160 NEXT L1
4170 POKE SID+1,0
4180 RETURN
```

Spectrum /Spectrum+

THE AUTO MECHANIC

```
  10 REM THE AUTO MECHANIC
  12 POKE 23658,8
  14 POKE 23609,40
  16 POKE 23692,255
  20 BORDER 1: PAPER 1: INK 7: C
LS
  30 PRINT INK 5;"SO YOU'RE HAVI
NG PROBLEMS WITH  YOUR CAR"
  32 BEEP .4,-4: BEEP .6,-12
  34 PAUSE 50
  40 PRINT INK 6;"LET'S SEE IF W
E CAN PIN DOWN THE TROUBLE..."
  42 BEEP .6,4: BEEP .4,12
  70 PRINT
  80 PRINT "A - ENGINE WON'T TUR
N OVER"
  90 PRINT "B - ENGINE TURNS, BU
T WON'T          START"
 100 PRINT "C - STARTS, THEN STA
LLS STRAIGHT      AWAY"
 110 PRINT "D - CAR RUNS, BUT ST
ALLS A LOT"
 120 PRINT "E - CAR RUNS, BUT ID
LES ROUGHLY"
 130 IF INKEY$<>"" THEN GO TO 13
0
 140 LET A$=INKEY$
 150 IF A$<"A" OR A$>"E" THEN GO
 TO 140
 160 GO SUB 1560
 170 IF A$="A" THEN GO TO 200
 172 IF A$="B" THEN GO TO 580
 174 IF A$="C" THEN GO TO 1200
```

```
 176 IF A$="D" THEN GO TO 1230
 178 IF A$="E" THEN GO TO 1350
 180 STOP
 190 REM ************************
 200 REM WON'T TURN OVER
 210 PRINT "WE'LL START BY CHECK
ING THE       BATTERY."
 220 PRINT "TURN ON THE LIGHTS."
 230 PRINT TAB 6; INK 6;"ARE THE
Y DIM?"
 240 GO SUB 1520
 250 IF A$="N" THEN GO TO 350: R
EM BATTERY OK
 260 PRINT INK 6;"ARE BETTERY CA
BLES LOOSE OR       CORRODED?"
 270 GO SUB 1520
 280 IF A$="Y" THEN PRINT "TIGHT
EN AND CLEAN"
 290 GO SUB 1580
 300 PRINT INK 6;"IS THE FAN BEL
T LOOSE?"
 310 GO SUB 1520
 320 IF A$="Y" THEN PRINT TAB 4;
"TIGHTEN THE FAN BELT"
 330 GO SUB 1580
 340 PRINT "JUMPER LEADS OR A PU
SH SHOULD BE ENOUGH TO START THE
 CAR": STOP
 350 PRINT INK 6;"IS THERE LOOSE
NESS OR CORROSION"
 360 PRINT INK 6;" AT THE STARTE
R END OF THE           BATTERY CABL
E?"
 370 GO SUB 1520
 380 IF A$="Y" THEN PRINT "TIGHT
EN AND CLEAN THE              CONN
ECTIONS"
 390 GO SUB 1580
 400 PRINT "PUT A PIECE OF METAL
 ACROSS THE"
 410 PRINT " SOLENOID TERMINALS.
"; INK 6;"DOES              STARTER
WORK?"
 420 GO SUB 1520
 430 IF A$="Y" THEN PRINT "THE I
GNITION SWITCH IS PROBABLY  FAUL
TY"
 440 GO SUB 1580
```

```
 450 IF A$="Y" THEN PRINT TAB 4;
"IT SHOULD BE REPLACED": STOP
 460 PRINT INK 6;"DOES STARTER C
LICK?"
 470 GO SUB 1520
 480 IF A$="Y" THEN PRINT "THE S
TARTER MAY BE JAMMED": GO TO 520
 490 PRINT "AS NOTHING HAPPENED,
 THE"
 500 PRINT " SOLENOID IS PROBABL
Y FAULTY. A"
 510 PRINT "  PUSH MAY START YOU
R CAR.": STOP
 520 REM STARTER JAMMED
 530 PRINT "TURN THE IGNITION OF
F, AND PUT"
 540 PRINT " THE CAR INTO A HIGH
 GEAR. PUSH"
 550 PRINT "  CAR FOR A FOOT OR
SO TO POP      STARTER LOOSE.":
STOP
 560 RETURN
 570 REM ************************
 580 REM TURN OVER, WON'T START
 590 PRINT "CHECK WIRES AT POINT
S FOR         DAMPNESS."
 600 PRINT INK 6;"IS THERE A CHA
NCE AREAS ARE        DAMP?"
 610 GO SUB 1520
 620 IF A$="Y" THEN PRINT "SPRAY
 WITH MOISTURE REMOVAL        SPRA
Y": GO SUB 1580
 630 PRINT INK 6;"IS THERE DUST
VISIBLE ON THE"
 640 PRINT INK 6;" INSULATING PA
RT OF THE COIL, OR"
 650 PRINT INK 6;"  ON THE DISTR
IBUTOR CAP?"
 660 GO SUB 1520
 670 IF A$="Y" THEN PRINT "WIPE
COIL SECTION AS WELL AS"
 680 IF A$="Y" THEN PRINT " INSI
DE AND OUTSIDE OF CAP.": GO SUB
1580
 690 PRINT "ENSURE ALL WIRES ARE
 TIGHT AND    DRY BEFORE CONTINUI
NG"
 700 GO SUB 1580
 710 PRINT "IF CAR STILL DOES NO
```

```
T START, IT  IS TIME TO CHECK TH
E SPARK       PLUGS:"
 720 GO SUB 1580
 730 PRINT INK 6;"DOES SPARK FRO
M THE END OF THE   PLUG WIRE JUM
P 3/8 INCH OR       MORE?"
 740 GO SUB 1520
 750 IF A$="Y" THEN PRINT "THE P
LUGS ARE FAULTY": GO SUB 1580
 760 IF A$="N" THEN GO TO 830
 770 PRINT INK 6;"ARE PLUGS GREA
SY?"
 780 GO SUB 1520
 790 IF A$="Y" THEN PRINT "AN EM
ERGENCY REPAIR CANNOT BE"
 800 IF A$="Y" THEN PRINT " MADE
 WHILE THE PLUGS ARE IN      THA
T STATE.": GO SUB 1580: GO TO 77
0
 810 IF A$="N" THEN PRINT "IF TH
IS IS AN EMERGENCY, TRY"
 820 IF A$="N" THEN PRINT " CLOS
ING THE GAP TO ABOUT HALF    NOR
MAL": GO SUB 1580: GO TO 910
 830 PRINT INK 6;"IS THERE ANY S
PARK AT ALL?"
 840 GO SUB 1520
 850 IF A$="Y" THEN GO TO 770
 860 GO SUB 870: STOP
 870 PRINT "CHECK ROTOR, COIL AN
D"
 880 PRINT " DISTRIBUTOR CAP FOR
 CRACKS. IF"
 890 PRINT "  THERE AREN'T ANY T
HERE, IT       LOOKS AS IF THE P
OINTS OR        CONDENSER IS YOU
R PROBLEM"
 900 PRINT "A REPAIR MAY WELL BE
 NEEDED": RETURN
 910 PRINT INK 6;"DO YOU HAVE GA
S IN THE TANK?"
 920 GO SUB 1520
 930 IF A$="N" THEN PRINT "FILL
TANK AND TRY AGAIN": GO SUB 1580
 940 PRINT INK 6;"ARE ALL FUEL A
ND VACUUM LINES     SECURE?"
 950 GO SUB 1520
 960 IF A$="N" THEN PRINT "ATTEN
D TO THESE AND TRY AGAIN": GO SU
```

```
B 1580
 970 PRINT "REMOVE THE AIR CLEAN
ER FROM THE"
 980 PRINT " CARBURETOR": GO SUB
 1580
 990 PRINT INK 6;"DOES IT LOOK D
RY?"
1000 GO SUB 1520
1010 IF A$="N" THEN GO TO 1080
1020 PRINT "TURN THE ENGINE OVER
 A FEW TIMES"
1030 PRINT " WITH YOUR HAND SEAL
ING THE AIR   INTAKE": GO SUB 15
80
1040 PRINT INK 6;"IS YOUR HAND W
ET WITH GAS?"
1050 GO SUB 1520
1060 IF A$="N" THEN PRINT "UNSCR
EW GAS CAP IN CASE AIR VENT"
1070 IF A$="N" THEN PRINT " IS P
LUGGED, THE FUEL PUMP MAY    NOT
 BE WORKING": GO SUB 900: STOP
1080 PRINT INK 6;"HAVE YOU BEEN
CRANKING THE"
1090 PRINT INK 6;" STARTER A LOT
 IN THE PAST FEW     MINUTES?"
1100 GO SUB 1520
1110 IF A$="N" THEN GO TO 1150
1120 PRINT "WAIT FOR A MINUTE OR
 SO, THEN"
1130 PRINT " HOLD THE GAS PEDAL
STEADILY ON"
1140 PRINT "  THE FLOOR WITHOUT
PUMPING IT.    THIS SHOULD GET Y
OU GOING": STOP
1150 PRINT "THE ENGINE MAY WELL
BE FLOODED"
1160 PRINT " AT THIS POINT AND T
HE FLOAT       VALVE STUCK OPEN."
: GO SUB 1580
1170 PRINT "TAP THE SIDE OF THE
CARBURETOR"
1180 PRINT " THEN TRY THE STARTI
NG PROCESS     AGAIN": STOP
1190 REM ***********************
1200 REM STARTS, THEN STALLS
1210 PRINT "THIS SUGGESTS A FAUL
TY BALLAST   RESISTOR, WHICH SHO
ULD BE           REPLACED": STOP
1220 REM ***********************
```

```
1230 REM RUNS,STALLS A LOT
1240 PRINT "THE PROBLEM IS EITHE
R CAUSED BY:"
1250 PRINT "  SHORTING (OR LOOSE
) WIRES;"
1260 PRINT "  A WEAK SPARK; OR"
1270 PRINT "  A FAULT IN THE FUE
L SYSTEM"
1280 PRINT "CHECK FIRST FOR LOOS
E OR SHORT-  ING WIRES.": GO SUB
 1580
1290 PRINT "IF THEY ARE NOT OK,
REPAIR. IF   THEY ARE, IT COULD
BE THE SPARK  PLUGS"
1300 GO SUB 870
1310 PRINT "THERE IS A FINAL CHE
CK WE CAN"
1320 PRINT " TRY ON YOUR SPARK P
LUGS": GO SUB 1580
1330 GO TO 1360
1340 REM **********************
1350 REM RUNS, ROUGH IDLE
1360 PRINT "IT COULD WELL BE THA
T ONE OR"
1370 PRINT " MORE OF YOUR SPARK
PLUGS ARE      FAULTY.": GO SUB 1
580
1380 PRINT "DISCONNECT THEM ONE
AT A TIME."
1390 PRINT " THE ONES WHICH DO N
OT CAUSE THE"
1400 PRINT "  ENGINE IDLE TO DRO
P ARE"
1410 PRINT "   FAULTY. CHECK THE
SE, THEN        RETURN TO SYSTEM
.": GO SUB 1580
1420 PRINT INK 6;"DID TEST SHOW
ANY PLUGS WERE      FAULTY?"
1430 GO SUB 1520
1440 IF A$="Y" THEN PRINT "REPLA
CE ALL PLUGS IF YOU CAN, OR"
1450 IF A$="Y" THEN PRINT " JUST
 THE ONES WHICH TESTED       FAU
LTY": GO TO 1580
1460 PRINT "THE MOST COMMON CAUS
E OF A BAD"
1470 PRINT " IDLE, ASSUMING THAT
 THE PLUGS"
1480 PRINT "  ARE OK, IS THAT YO
UR GAS MIX-"
```

```
1490 PRINT "   TURE IS SET TOO R
ICH, SO YOU     SHOULD ADJUST TH
IS": STOP
1500 PRINT "ADJUST THE IDLE SPEE
D-SCREW ON   THE THROTTLE LINKAG
E": STOP
1510 REM ***********************
1520 PRINT TAB 10; INVERSE 1; IN
K 6;"(Y - YES, N - NO)?"
1530 IF INKEY$<>"" THEN GO TO 15
30
1540 LET A$=INKEY$
1550 IF A$<>"Y" AND A$<>"N" THEN
 GO TO 1540
1560 PRINT TAB 12;"> OK   ";A$;"
 <"
1570 BEEP .03,12: BEEP .03,9: BE
EP .03,15: BEEP .03,18
1580 PAUSE 100
1590 PRINT
1595 POKE 23692,255
1600 RETURN
```

MEDICI

```
  10 REM MEDICI - PERSONAL CHECK
UP
  15 POKE 23658,8
  20 BORDER 1: PAPER 1: INK 7: C
LS
  30 GO SUB 1250
  40 PRINT "A - I AM BADLY OVERW
EIGHT"
  50 PRINT "B - I AM FAIRLY OVER
WEIGHT"
  60 PRINT "C - I AM SLIGHTLY OV
ERWEIGHT"
  70 PRINT "D - MY WEIGHT IS ABO
UT RIGHT"
  80 PRINT "E - I AM THINNER THA
N I SHOULD      BE"
  90 GO SUB 1170
 100 LET WEIGHT=5*(CODE (A$)-68)
: IF A$="E" THEN LET WEIGHT=0
 110 PRINT WEIGHT
```

```
 120 GO SUB 1250
 130 INK 5: PRINT "I ENGAGE IN E
XERCISE, THAT"
 140 PRINT "RAISES MY HEARTBEAT
TO 120 OR    MORE,";
 150 PRINT " FOR AT LEAST THE FO
LLOWINGNUMBER";
 160 PRINT " OF HOURS A WEEK:"
 170 INK 7: PRINT
 180 PRINT "A - LESS THAN A QUAR
TER"
 190 PRINT "B - MORE THAN A QUAR
TER, UP TO       THREE-QUARTERS"
 200 PRINT "C - FROM THREE QUART
ERS OF AN        HOUR UP TO ONE A
ND A HALF"
 210 PRINT "D - FROM ONE AND A H
ALF TO TWO       AND A HALF"
 220 PRINT "E - MORE THAN TWO AN
D A HALF         HOURS"
 230 GO SUB 1170
 240 LET EXERCISE=5*(CODE A$-63)
-5: IF A$="A" THEN LET EXERCISE=
0
 250 PRINT EXERCISE
 260 GO SUB 1250
 270 PRINT INK 5;"WHEN DRIVING:"
: PRINT
 280 PRINT "A - I HARDLY EVER WE
AR A SEAT        BELT"
 290 PRINT "B - I WEAR A SEAT BE
LT AROUND A      QUARTER OF THE T
IME"
 300 PRINT "C - I WEAR A SEAT BE
LT EVERY         SECOND JOURNEY"
 310 PRINT "D - I WEAR A SEAT BE
LT FOR           MOST, BUT NOT AL
L TRIPS"
 320 PRINT "E - I ALWAYS WEAR A
SEAT BELT"
 330 GO SUB 1170
 340 LET SEATBELT=2*(CODE A$-65)
 350 PRINT SEATBELT
 360 GO SUB 1250
 370 PRINT INK 5;"I AM CONSCIOUS
 OF NUTRITION AND TRY TO EAT HEA
LTHILY:"
 380 PRINT
 390 PRINT "A - ALL THE TIME"
```

```
 400 PRINT "B - NEARLY ALL THE T
IME"
 410 PRINT "C - A FAIR PROPORTIO
N OF THE         TIME"
 420 PRINT "D - FROM TIME TO TIM
E"
 430 PRINT "E - HARD'   AT ALL"
 440 GO SUB 1170
 450 LET DIET=-CODE     69
 460 PRINT DIET
 470 GO SUB 1250
 480 PRINT INK 5;"SMOKING (A CIG
AR COUNTS AS A     CIGARETTE)"
 490 PRINT
 500 PRINT "A - NOT AT ALL"
 510 PRINT "B - LESS THAN 15 CIG
ARETTES A        DAY"
 520 PRINT "C - 15 TO 25 CIGARET
TES A DAY"
 530 PRINT "D - 26 TO 42 CIGARET
TES A DAY"
 540 PRINT "E - MORE THAN 42 CIG
ARETTES A        DAY"
 550 GO SUB 1170
 560 LET SMOKING=-7*(CODE A$-65)
 570 PRINT SMOKING
 580 GO SUB 1250
 590 PRINT INK 5;"ALCOHOL - HOW
MANY DRINKS (ON    AVERAGE) DO YO
U HAVE EACH DAY?"
 600 PRINT
 610 PRINT "A - NONE"
 620 PRINT "B - LESS THAN 3"
 630 PRINT "C - 3 TO 6"
 640 PRINT "D - 7 TO 9"
 650 PRINT "E - MORE THAN 9"
 660 GO SUB 1170
 670 LET DRINK=-30
 680 IF A$="A" THEN LET DRINK=0
 690 IF  $="B" THEN LET DRINK=1
 700 IF A$="C" THEN LET DRINK=DR
INK/5
 710 IF A$="D" THEN LET DRINK=DR
INK/2
 720 PRINT DRINK
 730 GO SUB 1250
 740 PRINT INK 5;"IN GENERAL, HO
  STRESSFUL WOULD YOU SAY";
 750 PRINT INK 5;" YOUR LIFE HAS
 BEEN IN    THE PAST SIX MONTHS"
```

```
 760 PRINT
 770 PRINT "A - EXTREMELY STRESS
FUL"
 780 PRINT "B - FAIRLY STRESSFUL
"
 790 PRINT "C - SLIGHTLY STRESSF
UL"
 800 PRINT "D - NEUTRAL"
 810 PRINT "E - NOT STRESSFUL"
 820 GO SUB 1170
 830 LET STRESS=INT (2.5*(CODE A
$-69))
 840 PRINT STRESS
 850 GO SUB 1300: CLS
 860 PRINT "PERSONAL ASSESSMENT
FROM MEDICI:"
 870 PRINT : INK 6
 880 PRINT TAB 8;"WEIGHT:",WEIGH
T
 890 PRINT TAB 6;"EXERCISE:",EXE
RCISE
 900 PRINT TAB 4;"CAR SAFETY:",S
EATBELT
 910 PRINT TAB 5;"NUTRITION:",DI
ET
 920 PRINT TAB 7;"SMOKING:",SMOK
ING
 930 PRINT TAB 7;"ALCOHOL:",DRIN
K
 940 PRINT TAB 8;"STRESS:",STRES
S
 950 GO SUB 1300
 960 LET ANT=WEIGHT+EXERCISE+SEA
TBELT+DIET+SMOKING+DRINK+STRESS
 970 GO SUB 1300: PRINT
 980 PRINT INK 5;"YOUR RAW RATIN
G IS ";ANT: PRINT
 990 INK 7: PRINT "ON A SCALE WH
ERE ZERO IS""AVERAGE,";
1000 PRINT " THE LOWEST RATING I
S    BELOW -80, AND";
1010 PRINT " THE HIGHEST IS""OV
ER 30"
1020 GO SUB 1300: PRINT
1030 IF ANT<6 AND ANT>-6 THEN LE
T A$="AVERAGE": LET L$="62 TO 73
    72 TO 78"
1040 IF ANT<-5 AND ANT>-21 THEN
LET A$="BELOW AVERAGE": LET L$="
60 TO 66    65 TO 71"
```

```
1050 IF ANT<-20 THEN LET A$="POO
R": LET L$="60 OR LESS 65 OR LES
S"
1060 IF ANT<-45 THEN LET A$="VER
Y POOR"
1070 IF ANT<-60 THEN LET A$="VER
Y, VERY POOR"
1080 IF ANT>5 AND ANT<15 THEN LE
T A$="GOOD": LET L$="74 TO 80
79 TO 85"
1090 IF ANT>14 THEN LET A$="EXTR
EMELY GOOD": LET L$="81 PLUS
86 PLUS"
1100 PRINT "THIS INDICATES YOUR
HEALTH       STATUS IS ";A$
1110 FOR J=1 TO 500: NEXT J: CLS
 : PRINT : PRINT
1120 PRINT TAB 8;"LIFE EXPECTANC
Y:"
1130 PRINT ''TAB 8; INK 5;"MALE
      FEMALE"
1140 PRINT 'TAB 6; INK 6;L$
1150 STOP
1160 REM ************
1170 REM ACCEPT INPUT
1180 IF INKEY$<>"" THEN GO TO 11
80
1190 LET A$=INKEY$
1200 IF A$<"A" OR A$>"E" THEN GO
 TO 1190
1210 PRINT : PRINT INK 6;"OK, ";
A$;"   ";
1220 BEEP .4,5*(CODE A$-65): RET
URN
1230 REM ***************
1240 REM DELAY/SPACE OUT
1250 FOR J=1 TO 200: NEXT J
1260 CLS
1270 PRINT
1280 PRINT INK 6;"WHICH OF THE F
OLLOWING IS          CLOSEST TO THE
 TRUTH (SELECT      ONE):"
1290 PRINT
1300 FOR J=1 TO 100: NEXT J
1310 BEEP .1,15: BEEP .1,20: RET
URN
```

FUZZY RITA

```
  10 REM FUZZY RITA
  20 GO SUB 1380: REM INITIALISE
  30 GO SUB 1160: REM OUTPUT OPT
IONS
  40 GO SUB 1250: REM DISCRIMINA
TION OPTIONS
  50 GO SUB 140: REM QUESTION US
ER
  60 GO SUB 480: REM MAKE DECISI
ON AND UPDATE KNOWLEDGE BASE
  70 PRINT INK 6;TAB 3;"PRESS <E
NTER> TO CONTINUE"
  80 IF X$<>"" THEN INPUT LINE I
$: GO TO 50
  90 PRINT INK 6;TAB 6;"TRAINING
"
 100 PRINT INK 6;TAB 3;"OR ANY K
EY THEN <ENTER> TO";TAB 12;"USE
RITA"
 110 INPUT LINE X$: GO TO 50
 120 STOP
 130 REM ************************
 140 REM QUESTION USER
 150 CLS
 160 PRINT "HERE ARE THE SUBJECT
S I CAN      DISCRIMINATE BETWEEN
:"
 170 PRINT
 180 FOR J=1 TO TT
 190 PRINT INK 5;" >  "; INK 7;A
$(J)
 200 NEXT J
 210 GO SUB 1500
 220 IF X$="" THEN PRINT INK 6;"
THINK OF ONE, THEN PRESS <ENTER>
"
 230 IF X$<>"" THEN PRINT INK 6;
"I AM READY NOW TO DETERMINE
 WHICH ONE YOU HAVE"
 240 IF X$="" THEN INPUT LINE J$
 250 LET ADD=.5
 260 FOR J=1 TO DQ
 270 LET ADD=ADD+ADD
 280 GO SUB 1500
```

```
 290 IF X$<>"" AND TT>2 THEN GO
TO 390: REM CHECK IF QUESTION CA
N BE JUMPED
 300 PRINT " ENTER A NUMBER FROM
"
 310 PRINT "1 (TRUE) TO 0 (FALSE
) ($ TO END)"
 320 PRINT : PRINT INK 5;E$(J)
 330 INPUT LINE H$: IF H$="$" TH
EN PRINT : PRINT "THANK YOU": PR
INT : STOP
 340 LET C(J)=VAL H$
 350 LET C(J)=ADD*C(J)
 360 NEXT J
 370 RETURN
 380 REM ***********************
 390 REM CHECK IF QUESTION CAN B
E JUMPED
 400 LET JUMP=1
 410 FOR W=1 TO TT
 420 IF ABS (B(W,J)-B(1,J))>.7 T
HEN LET JUMP=0
 430 NEXT W
 440 IF JUMP=0 THEN GO TO 300
 450 LET C(JUMP)=B(W,J)
 460 GO TO 360
 470 REM ***********************
 480 REM MAKE DECISION
 490 FOR J=1 TO TT
 500 LET D(J)=0: LET E(J)=0: LET
 F(J)=0
 510 NEXT J
 520 LET ADD=.5
 530 FOR J=1 TO TT
 540 LET ADD=ADD+ADD
 550 FOR X=1 TO DQ
 560 REM PLAY WITH VALUES IN NEX
T THREE LINES FOR MOST EFFICIENT
 RESULTS
 570 IF C(X)=B(J,X) THEN LET D(J
)=D(J)+1
 580 IF ABS (C(X)-B(J,X))<.6*ADD
 THEN LET E(J)=E(J)+.4
 590 IF ABS (C(X)-B(J,X))<1.2*AD
D THEN LET F(J)=F(J)+.1
 600 NEXT X
 610 NEXT J
 620 LET A1=1: LET A2=1: LET A3=
1
```

```
 630 LET F1=1: LET F2=1: LET F3=
1
 640 FOR J=1 TO TT
 650 IF D(J)>F1 THEN LET F1=D(J)
: LET A1=J
 660 IF E(J)>F2 THEN LET F2=E(J)
: LET A2=J
 670 IF F(J)>F3 THEN LET F3=F(J)
: LET A3=J
 680 NEXT J
 690 REM ** ANNOUNCE RESULT **
 700 PRINT
 710 LET CFLG=0
 720 PRINT "THE MOST LIKELY RESU
LT IS"'TAB 1; INK 5;A$(A1)
 730 IF A2<>A1 THEN PRINT "THE N
EXT MOST LIKELY IS"'TAB 1; INK 5
;A$(A2): LET CFLG=1
 740 IF A3<>A2 AND A3<>A1 THEN P
RINT "THE NEXT MOST LIKELY IS"'T
AB 1; INK 5;A$(A3): LET CFLG=2
 750 PRINT
 760 PRINT INK 6;"IS THE MOST LI
KELY RESULT          CORRECT?  (Y O
R N)"
 770 INPUT LINE F$
 780 IF F$<>"Y" AND F$<>"N" THEN
 GO TO 770
 790 IF F$="Y" AND X$<>"" THEN R
ETURN
 800 IF F$="Y" THEN GO TO 980
 810 IF TT=2 AND A1=1 THEN LET A
1=2: GO TO 980
 820 IF TT=2 THEN LET A1=1: GO T
O 980
 830 IF CFLG=0 THEN GO TO 890
 840 PRINT "IS MY SECOND CHOICE
CORRECT?            (Y OR N)"
 850 INPUT LINE F$
 860 IF F$="N" THEN GO TO 890
 870 IF CFLG=1 THEN LET A1=A2: G
O TO 980
 880 IF CFLG=2 THEN LET A1=A3: G
O TO 980
 890 GO SUB 1500
 900 FOR J=1 TO TT
 910 PRINT INK 5;J;"- "; INK 7;A
$(J)
 920 NEXT J
```

```
 930 PRINT
 940 PRINT "WHICH IS THE CORRECT
 ONE?"
 950 INPUT A1
 960 IF A1<1 OR A1>TT THEN GO TO
 950
 970 REM ** EDUCATING RITA **
 972 REM (UPDATE KNOWLEDGE BASE)
 980 FOR J=1 TO DQ
 990 IF B(A1,J)<>0 THEN LET B(A1
,J)=(C(J)+5*B(A1,J))/6
1000 IF B(A1,J)=0 THEN LET B(A1,
J)=C(J)
1010 LET B(A1,J)=INT (.5+(10*B(A
1,J))/10)
1020 NEXT J
1030 PRINT
1040 IF U$="" THEN RETURN
1050 FOR J=1 TO TT
1060 PRINT : GO SUB 1500
1070 PRINT A$(J)
1080 PRINT
1090 FOR K=1 TO DQ
1100 PRINT E$(K); INK 5;B(J,K)
1110 NEXT K
1120 NEXT J
1130 PRINT
1140 RETURN
1150 REM ***********************
1160 REM OUTPUT OPTIONS
1170 LET TT=0
1180 LET TT=TT+1
1190 GO SUB 1500
1200 PRINT INK 6;"ENTER OUTPUT O
PTION NUMBER ";TT'" (PRESS <ENTE
R> TO END)"
1210 INPUT LINE A$(TT)
1215 PRINT : PRINT A$(TT)
1220 IF A$(TT,1)=" " OR TT=51 TH
EN LET TT=TT-1: RETURN
1230 GO TO 1180
1240 REM ***********************
1250 REM DISCRIMINATION QUESTION
S
1260 CLS
1270 FOR J=1 TO TT
1280 PRINT A$(J)
1290 NEXT J
1300 LET DQ=0
```

```
1310 LET DQ=DQ+1
1320 GO SUB 1500
1330 PRINT INK 6;"ENTER QUESTION
 NUMBER ";DQ'" (PRESS <ENTER> TO
 END)"
1340 INPUT LINE E$(DQ)
1345 PRINT : PRINT E$(DQ)
1350 IF E$(DQ,1)=" " OR DQ=51 TH
EN LET DQ=DQ-1: RETURN
1360 GO TO 1310
1370 REM ************************
1380 REM INITIALISATION
1382 POKE 23658,8
1384 POKE 23609,40
1386 POKE 23692,255
1390 BORDER 1: PAPER 1: INK 7: C
LS
1400 REM REDUCE ARRAYS IN NEXT L
INE IN ACCORDANCE WITH YOUR NEED
S
1410 DIM A$(50,15): DIM B(50,50)
: DIM C(50): DIM D(50): DIM E$(5
0,40): DIM E(50): DIM F(50)
1420 LET X$=""
1430 PRINT "PRESS ANY KEY, THEN
<ENTER> IF"
1440 PRINT "YOU WANT TO SEE THE
UPDATED"
1450 PRINT "KNOWLEDGE BASE AFTER
 EACH RUN;"
1460 PRINT "JUST PRESS <ENTER> I
F YOU DON'T"
1470 INPUT LINE U$
1480 CLS
1490 RETURN
1500 PRINT INK 2;"--------------
------------------"
1502 POKE 23692,255
1510 RETURN
```

HASTE

```
  5 PAPER 1: INK 7: BORDER 1
 10 REM HASTE
 15 POKE 23609,40: POKE 23658,8
```

```
  20 DIM A$(255,20): DIM B$(255,
50)
  30 LET F$="": LET K=0: LET F=0
  CLS
  40 REM **************
  50 LET FLAG=0
  60 INPUT "> "; LINE D$: PRINT
D$: LET K=K+1
  70 IF D$="" THEN STOP
  80 IF D$( TO 1)="?" THEN GO TO
 200
  90 LET E=0
 100 LET E=E+1
 110 IF D$(E TO E)="*" THEN GO T
O 140
 120 IF E<LEN D$ THEN GO TO 100
 130 PRINT "INVALID ENTRY": BEEP
 .5,0: GO TO 50
 140 IF FLAG=3 THEN RETURN
 150 LET F=F+1: IF F=256 THEN ST
OP
 160 LET A$(F)=D$( TO E-1)+"@"
 170 LET B$(F)=D$(E+1 TO )+"@"
 175 BEEP .1,10: BEEP .3,15
 180 GO TO 50
 190 REM **************
 200 REM INTERROGATE
 210 LET FLAG=4: LET TRUE=0
 220 IF D$(LEN D$ TO )="/" THEN
LET FLAG=3
 230 FOR J=1 TO (LEN D$)-5
 240 IF D$(J TO J+4)=" AND " THE
N LET FLAG=5: LET TRUE=J
 250 NEXT J
 260 IF FLAG=5 THEN GO TO 410
 270 IF D$( TO 3)="?/*" THEN LET
 FLAG=1
 280 IF D$( TO 4)="?/*/" THEN LE
T FLAG=2
 290 IF FLAG=3 THEN GO SUB 90: L
ET F$=D$(2 TO E-1)
 300 IF FLAG=1 THEN LET F$=D$(4
TO )
 310 LET E=0
 320 LET E=E+1
 330 IF CODE A$(E)=32 AND FLAG=4
 AND TRUE=0 THEN PRINT "FALSE":
BEEP .1,10: BEEP .3,15
```

```
 340 IF CODE A$(E)=32 THEN GO TO
 520
 345 LET P$=A$(E): LET Q$=B$(E):
 GO SUB 600: GO SUB 650
 350 IF FLAG=4 AND "?"+P$+"*"+Q$
=D$ THEN PRINT "TRUE": BEEP .1,1
0: BEEP .3,15: LET TRUE=1
 360 IF FLAG=3 AND F$=P$ THEN PR
INT Q$
 370 IF FLAG=2 THEN PRINT P$;"*"
;Q$
 380 IF FLAG=1 AND F$=Q$ THEN PR
INT P$
 385 BEEP .05,0
 390 IF E<K THEN GO TO 320
 400 REM ***************
 410 LET F$=D$(4 TO TRUE-1): LET
 G$=D$(TRUE+7 TO )
 420 LET E=0
 430 LET E=E+1
 440 IF CODE A$(E)=32 THEN GO TO
 520
 450 LET Q$=B$(E): GO SUB 650: I
F Q$=F$ THEN GO TO 470
 460 IF E<K THEN GO TO 430
 470 LET H=0
 480 LET H=H+1
 490 IF CODE B$(H)=32 THEN GO TO
 460
 500 LET Q$=B$(H): GO SUB 650: L
ET P$=A$(E): GO SUB 600: LET W$=
P$: LET P$=A$(H): GO SUB 600: IF
 Q$=G$ AND W$=P$ THEN PRINT W$
 510 IF E<K THEN GO TO 480
 520 PRINT INK 6;TAB 5;" > END O
F ANSWER <"
 530 GO TO 50
 600 LET T=0
 610 LET T=T+1
 620 IF P$(T TO T)="@" THEN LET
P$=P$( TO T-1): RETURN
 630 GO TO 610
 650 LET T=0
 660 LET T=T+1
 670 IF Q$(T TO T)="@" THEN LET
Q$=Q$( TO T-1): RETURN
 675 BEEP .02,0
 680 GO TO 660
```

EASLE

```
  10 REM EASLE
  20 BORDER 1: PAPER 1: INK 7: C
LS
  25 POKE 23609,60
  27 POKE 23658,8
  30 DIM X(40): DIM Y(40): DIM Z
(40)
  40 PRINT
  50 INPUT "> "; LINE A$
  55 PRINT "> ";A$: BEEP .4,10
  60 IF A$="" THEN BEEP 1,-15: S
TOP
  70 REM ***********************
  80 FOR J=1 TO 40
  90 LET X(J)=0: LET Y(J)=0: LET
 Z(J)=0
 100 NEXT J
 110 REM ***********************
 120 LET R=0: LET S=0: LET T=0:
LET CFIRST=0: LET CSECND=0: LET
EDGE=0
 130 FOR J=1 TO LEN A$
 140 LET B$=A$(J)
 150 IF B$="(" THEN LET S=S+1: L
ET Z(S)=J: IF T=0 THEN LET CFIRS
T=J
 160 IF B$=")" THEN LET T=T+1: L
ET Y(T)=J: IF CSECND<>0 AND EDGE
=0 THEN LET EDGE=J-1
 170 IF T=1 AND B$=")" THEN LET
CSECND=J
 180 IF B$=" " THEN LET R=R+1: L
ET X(R)=J
 190 NEXT J
 200 IF S=T THEN GO TO 260: REM
( ) BALANCE
 210 BEEP .4,-10: IF S<T THEN PR
INT " -> MISSING  ("
 220 IF S>T THEN PRINT " -> MISS
ING  )"
 230 INPUT "   >    "; LINE B$
 235 PRINT "   >    ";B$
 240 LET A$=A$+B$
 250 GO TO 80
 260 LET FLAG=0
```

```
 270 IF A$( TO 5)="CAR (" THEN L
ET FLAG=1: GO SUB 420: GO TO 340
 280 IF A$( TO 5)="CDR (" THEN L
ET FLAG=2: GO SUB 470: GO TO 340
 290 IF A$( TO 6)="CONS (" THEN
LET FLAG=3: GO SUB 550: GO TO 34
0
 300 IF A$( TO 6)="ATOM (" THEN
LET FLAG=4: GO SUB 690: GO TO 34
0
 310 IF A$( TO 4)="EQ (" THEN LE
T FLAG=5: GO SUB 780: GO TO 340
 320 IF A$( TO 6)="NULL (" THEN
LET FLAG=6: GO SUB 920
 340 IF FLAG<>0 THEN GO SUB 360
 350 GO TO 40
 360 REM ** RETURN ANSWER **
 370 PRINT "    VALUE IS..."
 380 IF B$<>"()" THEN PRINT "
";B$
 390 IF B$="()" THEN PRINT "    N
IL"
 400 BEEP .3,20: RETURN
 410 REM **********************
 420 REM         **  CAR  **
 430 IF S=2 THEN LET B$=A$(Z(2)+
1 TO X(2))
 440 IF S>2 THEN LET B$=A$(CFIRS
T TO CSECND)
 450 RETURN
 460 REM **********************
 470 REM         **  CDR  **
 480 GO SUB 420
 490 LET LB=LEN B$+7
 495 LET KK=LB+EDGE-2: IF KK>LEN
 A$ THEN LET KK=LEN A$
 500 LET B$="("+A$(LB TO KK)
 510 IF B$(LEN B$-1 TO )="))" TH
EN LET B$=B$( TO LEN B$-1)
 520 IF B$(2)=" " THEN LET B$="(
"+B$(3 TO )
 530 RETURN
 540 REM **********************
 550 REM         **  CONS **
 560 LET B$=A$(7 TO )
 570 LET J=0
 580 IF B$(1)="(" THEN LET J=1
 590 LET J=J+1
 600 IF B$(J)="(" THEN GO TO 630
```

```
 610 IF J<LEN B$ THEN GO TO 590
 620 LET B$="    > CONS ERROR <":
RETURN
 630 LET LB=LEN B$-1
 640 LET B$="("+B$( TO J-1)+B$(J
+1 TO )
 650 LET B$=B$( TO LB)
 660 IF B$(LEN B$-1 TO )=" )" TH
EN LET B$=B$( TO LEN B$-2)+")"
 670 RETURN
 680 REM ***********************
 690 REM         **  ATOM **
 700 LET A$=A$(7 TO LEN A$-1)
 710 LET J=0: LET B$="NIL"
 720 LET J=J+1
 730 IF A$(J)=" " OR A$(J)="(" T
HEN RETURN
 740 IF J<LEN A$ THEN GO TO 720
 750 LET B$="T"
 760 RETURN
 770 REM ***********************
 780 REM         **   EQ   **
 790 LET A$=A$(5 TO )
 800 LET A$=A$( TO LEN A$-1)
 810 LET J=0: LET B$="NIL"
 820 LET J=J+1
 830 IF A$(J)=")" THEN RETURN
 840 IF A$(J)=" " THEN GO TO 870
 850 IF J<LEN A$ THEN GO TO 820
 860 RETURN
 870 LET C$=A$( TO J-1)
 880 LET A$=A$(J+1 TO )
 890 IF C$=A$ THEN LET B$="T"
 900 RETURN
 910 REM ***********************
 920 REM         **  NULL **
 930 LET B$="NIL"
 940 IF A$="NULL ()" THEN BEEP .
4,-20: PRINT "ILLEGAL - NULL NEE
DS ARGUMENT": LET B$=""
 950 IF A$="NULL (())" THEN LET
B$="T"
 960 RETURN
```

PROLOG-A

```
    5 PAPER 1: BORDER 1: INK 7
   10 REM PROLOG-A (SIMPLE FRONT-
END)
   15 POKE 23609,40: POKE 23658,8
:
   20 REM * ALL INPUT IN UPPER CA
SE *
   30 GO TO 50
   40 PRINT "NO (MORE) ANSWERS":
RETURN
   50 GO SUB 3270: REM INITIALISE
   60 REM ******************
   70 PRINT
   75 PRINT AT 0,0;"
                      "

   80 INPUT "&."; LINE J$
   90 IF J$="" THEN STOP
   92 CLS
   95 PRINT INK 7; PAPER 0; FLASH
 1;AT 0,10;"PLEASE WAIT"
  100 IF J$="LIST ALL" THEN GO SU
B 1860: GO TO 70
  110 IF J$( TO 5)="LIST " THEN L
ET J$=J$(5 TO )+" ": GO SUB 990:
 GO TO 70
  120 IF J$(LEN J$ TO )<>")" THEN
 PRINT "1.": INPUT L$: LET J$=J$
+L$: GO TO 120
  130 LET LJ=LEN J$
  140 LET J$=J$( TO LJ-1)+" ": RE
M STRIP FINAL ), REPLACE WITH S
PACE
  150 LET LJ=LEN J$
  160 LET FLAG=0
  170 IF J$( TO 4)="ADD(" THEN LE
T J$=J$(5 TO ): LET FLAG=1
  180 LET RULEFLAG=0: LET PLUSFLA
G=0: LET ARITHFLAG=0
  190 FOR R=1 TO LEN J$
  195 IF (R+3)>LEN J$ THEN GO TO
205
  200 IF J$(R TO R+3)=" IF " THEN
 LET RULEFLAG=R: LET FLAG=6
```

```
 205 IF (R+4)>LEN J$ THEN GO TO
215
 210 IF J$(R TO R+4)=" AND " THE
N LET PLUSFLAG=R
 215 IF (R+3)>LEN J$ THEN GO TO
225
 220 IF J$(R TO R+3)="SUM(" THEN
 LET ARITHFLAG=1
 225 IF (R+5)>LEN J$ THEN GO TO
235
 230 IF J$(R TO R+5)="TIMES(" TH
EN LET ARITHFLAG=2
 235 IF (R+5)>LEN J$ THEN GO TO
245
 240 IF J$(R TO R+5)=" LESS " TH
EN LET ARITHFLAG=3
 245 IF (R+2)>LEN J$ THEN GO TO
260
 250 IF J$(R TO R+2)="INT" THEN
LET ARITHFLAG=4
 260 NEXT R
 265 IF LEN J$<3 THEN GO TO 275
 270 IF J$( TO 3)="IS(" THEN LET
 J$=J$(4 TO ): LET FLAG=2
 275 IF LEN J$<10 THEN GO TO 285
 280 IF J$( TO 10)="WHICH(X : "
THEN LET J$=J$(11 TO ): LET FLAG
=3
 285 IF LEN J$<16 THEN GO TO 300
 290 IF J$( TO 16)="WHICH((X Y)
: X " THEN LET J$=J$(17 TO ): LE
T FLAG=4
 300 IF FLAG=0 THEN PRINT "SYNTA
X ERROR": GO TO 70
 310 LET LJ=LEN J$
 320 REM NOW SEND TO RELEVANT SU
BROUTINES
 330 IF PLUSFLAG<>0 THEN GO SUB
1950: GO TO 70: REM ENCODE RULE
CONTAINING AND
 340 IF RULEFLAG<>0 AND FLAG<>5
THEN GO SUB 1110: REM ENCODE RUL
E
 350 IF ARITHFLAG<>0 THEN GO SUB
 2430: GO TO 70: REM ARITHMETIC
 360 IF J$(LEN J$-2 TO )=" X " O
R J$(LEN J$-2 TO )=" Y " THEN LE
T [illegible]=J$( TO LJ-2)+" "
 [illegible] LET LJ=LEN J$
```

```
 380 IF FLAG=1 THEN GO SUB 440:
REM ADD
 390 IF FLAG=2 THEN GO SUB 520:
REM IS
 400 IF FLAG=3 THEN GO SUB 610:
REM WHICH
 410 IF FLAG=4 THEN GO SUB 830:
REM WHICH2
 420 GO TO 70
 430 REM ***********************
 440 REM           ADD
 450 LET K=0
 460 LET K=K+1
 470 IF CODE Z$(K)=32 THEN LET Z
$(K)=J$+"@": BEEP .1,10: BEEP .3
,15: RETURN
 480 IF K<500 THEN GO TO 460
 490 PRINT INK 5; FLASH 1;"MEMEO
RY FULL": BEEP .5,15
 500 RETURN
 510 REM ***********************
 520 REM           IS
 530 LET K=0
 540 LET K=K+1
 550 IF CODE Z$(K)=32 THEN GO TO
 580
 560 LET A$=Z$(K): GO SUB 8500:
LET J$=J$( TO LEN J$-1): IF A$=J
$+" " THEN PRINT INK 6;"YES": GO
 TO 590
 570 IF K<500 THEN GO TO 540
 580 PRINT INK 6;"NO"
 590 BEEP .1,10: BEEP .3,15: RET
URN
 600 REM ***********************
 610 REM           WHICH
 620 IF J$(1)="X" THEN GO TO 710
 630 LET J$=J$( TO LJ-1)
 640 LET K=0
 650 LET K=K+1
 660 IF CODE Z$(K)=32 THEN GO TO
 690
 670 LET A$=Z$(K): GO SUB 8500:
IF J$=A$( TO LEN J$) THEN PRINT
INK 6;A$(LEN J$ TO )
 680 IF K<500 THEN GO TO 650
 690 GO SUB 40
 695 BEEP .1,10: BEEP .3,15
 700 RETURN
 710 REM * QUERY STARTS WITH X *
```

```
 720 LET J$=J$(3 TO LEN J$)
 730 LET LJ=LEN J$
 740 LET K=0
 750 LET K=K+1
 760 IF CODE Z$(K)=32 THEN GO TO
 800
 770 LET A$=Z$(K): GO SUB 8500:
LET Q$=A$(LEN A$-LJ+1 TO )
 780 IF Q$=J$ THEN PRINT INK 6;A
$( TO LEN A$-LJ-1)
 790 IF K<500 THEN GO TO 750
 800 GO SUB 40
 805 BEEP .1,10: BEEP .3,15
 810 RETURN
 820 REM **********************
 830 REM          WHICH2
 840 LET J$=J$( TO LJ-2)
 850 LET LJ=LEN J$
 860 LET K=0
 870 LET K=K+1
 880 IF CODE Z$(K)=32 THEN GO TO
 960
 890 LET LFLAG=0
 895 LET A$=Z$(K)
 900 FOR L=1 TO 30-LJ
 910 IF A$(L TO L+LJ-1)=J$ THEN
LET LFLAG=L
 920 NEXT L
 930 IF LFLAG=0 THEN GO TO 950
 935 GO SUB 8500
 940 PRINT A$( TO LFLAG-2);A$(LF
LAG+LJ TO )
 945 BEEP .05,0
 950 IF K<500 THEN GO TO 870
 960 GO SUB 40
 965 BEEP .1,10: BEEP .3,15
 970 RETURN
 980 REM **********************
 990 REM            LIST
1000 LET K=0
1010 LET K=K+1
1020 IF CODE Z$(K)=32 THEN RETUR
N
1030 LET LFLAG=0
1040 FOR L=1 TO LEN Z$(K)-LEN J$
1050 LET A$=Z$(K): IF A$(L TO L+
LEN J$-1)=J$ THEN LET LFLAG=1
1060 NEXT L
1070 IF LFLAG=1 THEN LET A$=Z$(K
```

```
): GO SUB 8500: PRINT INK 6;A$
1075 BEEP .05,0
1080 IF K<500 THEN GO TO 1010
1090 RETURN
1100 REM ************************
1110 REM          FORM RULES
1120 LET R=RULEFLAG
1130 LET E$=J$( TO R): LET F$=J$
(R+4 TO )
1140 IF E$(1)<>"X" THEN PRINT IN
K 5; FLASH 1;"RULE ERROR": GO TO
 70
1150 REM NEXT LINE DETECTS INPUT
S LIKE      X EATS Y IF X IS-A Y
1160 IF F$(LEN F$-1 TO )="Y " TH
EN GO TO 1390
1170 PRINT INK 7;TAB 8;"COMPILIN
G RULE"
1180 FOR T=1 TO 100
1190 LET R$(T)=""
1200 NEXT T
1210 LET E$=E$(3 TO ): LET F$=F$
(3 TO LEN F$)
1220 LET K=0: LET RR=0
1230 LET K=K+1
1240 IF CODE Z$(K)=32 THEN GO TO
 1300
1245 LET A$=Z$(K): GO SUB 8500
1250 IF A$(LEN A$-LEN F$+1 TO )<
>F$ THEN GO TO 1370
1260 LET RR=RR+1
1270 LET R$(RR)=A$( TO LEN A$-LE
N F$)+E$+"@"
1280 PRINT "> ";: LET A$=R$(RR):
 GO SUB 8500: PRINT INK 6;A$
1285 BEEP .1,10: BEEP .3,15
1290 GO TO 1230
1300 IF RR=0 THEN RETURN
1310 LET RC=0
1320 LET RC=RC+1
1330 LET Z$(K)=R$(RC)
1340 IF K<500 THEN LET K=K+1
1350 IF RC<RR THEN GO TO 1320
1360 RETURN
1370 IF K<500 THEN GO TO 1230
1380 RETURN
1390 REM * RULE WITH 2 VARIABLES
 *
1400 FOR T=1 TO 100
```

```
1410 LET R$(T)=""
1420 NEXT T
1430 LET K=0: LET RR=0
1440 IF K=500 THEN RETURN
1450 LET K=K+1
1460 IF CODE Z$(K)=32 THEN GO TO
 1770
1470 REM SPLIT INTO THREE WORDS
1480 LET Q$=Z$(K)
1490 LET J=0
1500 LET J=J+1
1510 IF Q$(J TO J)=" " THEN GO T
O 1540
1520 IF J<LEN Q$ THEN GO TO 1500
1530 PRINT INK 5;"RULE COMPILING
 ERROR": GO TO 70
1540 LET A$=Q$( TO J)
1550 LET Q$=Q$(J+1 TO )
1560 LET J=0
1570 LET J=J+1
1580 IF Q$(J)=" " THEN GO TO 161
0
1590 IF J<LEN Q$ THEN GO TO 1570
1600 PRINT INK 5;"RULE COMPILING
 ERROR": GO TO 70
1610 LET B$=Q$( TO J)
1620 LET Q$=Q$(J+1 TO )
1630 LET J=0
1640 LET J=J+1
1650 IF Q$(J)=" " THEN GO TO 168
0
1660 IF J<LEN Q$ THEN GO TO 1640
1670 PRINT INK 5;"RULE COMPILING
 ERROR": GO TO 70
1680 PRINT INK 7;TAB 8;"COMPILIN
G RULE"
1690 LET C$=Q$( TO J)
1700 LET M$=F$(3 TO 3+LEN B$-1)
1710 IF B$<>M$ THEN GO TO 1440
1720 LET RR=RR+1
1730 LET N$=E$(3 TO LEN E$-2)
1740 LET R$(RR)=A$+N$+C$+"@"
1750 PRINT "> ";: LET A$=R$(RR):
 GO SUB 8500: PRINT INK 6;A$
1755 BEEP .1,10: BEEP .3,15
1760 GO TO 1440
1770 IF RR=0 THEN RETURN
1780 LET M=0
1790 LET M=M+1
```

```
1800 IF M>RR THEN RETURN
1810 LET Z$(K)=R$(M)
1820 IF K=500 THEN PRINT INK 5;
FLASH 1;"OUT OF MEMORY": BEEP .5
,15: GO TO 70
1830 LET K=K+1
1840 GO TO 1790
1850 REM ************************
1860 REM           LIST ALL
1870 PRINT
1880 LET K=0
1890 LET K=K+1
1900 IF CODE Z$(K)=32 THEN RETUR
N
1910 LET A$=Z$(K): GO SUB 8500:
PRINT INK 6;A$
1915 BEEP .05,0
1920 IF K<500 THEN GO TO 1890
1930 RETURN
1940 REM ************************
1950 REM FORM RULES WITH 'AND'
         OF THE FOLLOWING
         TYPE:
1960 REM (X EATS Y IF X IS-A
         BIRD AND Y COMES-IN
         BOXES)
1970 REM X STATEMENT MUST BE IN
         LIST PRECEDING Y FOR
         ALL EXAMPLES TO BE
         CODED
1980 REM SPLIT INTO SECTIONS
1990 LET J$=J$(2 TO ): REM STRIP
 "X"
2000 PRINT INK 7;TAB 8;"COMPILIN
G RULE"
2010 LET J=1
2020 LET J=J+1
2030 IF J$(J)=" " THEN GO TO 206
0
2040 IF J<LEN J$ THEN GO TO 2020
2050 PRINT INK 5;"RULE COMPILING
 ERROR": RETURN
2060 LET I$=J$( TO J): REM RELAT
IONSHIP 1
2070 LET J$=J$(J+7 TO ): REM
        STRIP TO START OF SECOND
        RELATIONSHIP
2080 LET J=1: LET COUNT=0
2090 LET J=J+1
```

```
2100 IF J$(J)=" " THEN LET COUNT
=COUNT+1
2110 IF COUNT=2 THEN GO TO 2140
2120 IF J<LEN J$ THEN GO TO 2090
2130 PRINT INK 5;"RULE COMPILING
 ERROR": RETURN
2140 LET B$=J$( TO J): REM STATE
MENT 1
2150 LET C$=J$(J+6 TO ): REM STA
TEMENT 2
2160 IF C$=" " THEN PRINT "RULE
COMPILING ERROR": RETURN
2170 REM NOW GO THROUGH DATABASE
2180 FOR T=1 TO 200
2190 LET R$(T)=""
2200 NEXT T
2210 LET R1=0: LET R2=99
2220 LET K=0
2230 LET K=K+1
2240 IF CODE Z$(K)=32 THEN GO TO
 2310
2250 IF R1=99 OR R2=200 THEN PRI
NT "MEMORY SHORTAGE": GO TO 2310
2260 LET LB=LEN B$
2270 LET A$=Z$(K): GO SUB 8500:
IF A$(LEN A$-LB+1 TO )=B$ THEN L
ET R1=R1+1: LET R$(R1)=A$( TO LE
N A$-LB)+"@"
2280 LET LC=LEN C$
2290 LET A$=Z$(K): GO SUB 8500:
IF A$(LEN A$-LC+1 TO )=C$ THEN L
ET R2=R2+1: LET R$(R2)=A$( TO LE
N A$-LC)+"@"
2300 IF K<500 THEN GO TO 2230
2310 IF CODE R$(100)=32 THEN PRI
NT INK 5;"STATEMENT 2 OF INPUT N
OT IN DATABASE": BEEP .1,10: BEE
P .1,15: RETURN
2320 REM NOW ENCODE RULES
2330 LET R1=0: LET R2=99
2340 LET R2=R2+1
2350 LET R1=R1+1
2360 IF R$(R1)=" " OR R$(R2)=" "
 THEN GO TO 2370
2362 GO SUB 8600: GO SUB 8700
2365 LET Z$(K)=W$+I$+V$+" "+"@"
2370 PRINT "> ";: LET A$=Z$(K):
GO SUB 8500: PRINT INK 6;A$
2375 BEEP .1,10: BEEP .3,15
```

```
2380 LET K=K+1
2390 IF CODE R$(R2+1)<>32 THEN G
O TO 2340
2400 IF CODE R$(R1+1)<>32 THEN G
O TO 2350
2410 RETURN
2420 REM ***********************
2430 REM          ARITHMETIC
2440 LET LJ=LEN J$
2450 IF ARITHFLAG<3 THEN GO SUB
2490
2460 IF ARITHFLAG=3 THEN GO SUB
2890
2470 IF ARITHFLAG=4 THEN GO SUB
3080
2480 RETURN
2490 REM ******SUM TIMES*******
2500 LET J$=J$(5 TO LJ)
2510 IF J$( TO 2)="S(" THEN LET
J$=J$(3 TO )
2520 LET LJ=LEN J$
2530 LET K=0
2540 LET K=K+1
2550 IF J$(K)=" " THEN LET A$=J$
( TO K-1): LET J$=J$(K+1 TO ): G
O TO 2580
2560 IF K<LJ THEN GO TO 2540
2570 PRINT INK 5;TAB 6;"ARITHMET
IC ERROR": RETURN
2580 LET LJ=LEN J$
2590 LET K=0
2600 LET K=K+1
2610 IF J$(K)=" " THEN LET B$=J$
( TO K-1): LET J$=J$(K+1 TO ): G
O TO 2640
2620 IF K<LJ THEN GO TO 2600
2630 PRINT INK 5;TAB 6;"ARITHMET
IC ERROR": RETURN
2640 LET LJ=LEN J$
2650 LET K=0
2660 LET K=K+1
2670 IF J$(K)=")" THEN LET C$=J$
( TO K-1): GO TO 2700
2680 IF K<LJ THEN GO TO 2660
2690 PRINT INK 5;TAB 2;"ERROR [T
OO MANY VARIABLES]": RETURN
2700 LET AN=0: LET BN=0: LET CN=
0
2710 IF CODE A$>57 THEN LET AN=1
```

```
2720 IF CODE B$>57 THEN LET BN=2
2730 IF CODE C$>57 THEN LET CN=4
2740 LET GUIDE=AN+BN+CN: IF GUID
E=3 OR GUIDE=5 OR GUIDE=6 THEN G
O TO 2690
2750 IF ARITHFLAG=2 THEN GO TO 2
820: REM TIMES
2760 IF GUIDE>0 THEN GO TO 2790
2770 IF VAL A$+VAL B$=VAL C$ THE
N PRINT INK 6;"YES": BEEP .1,10:
 BEEP .3,15: RETURN
2780 PRINT INK 6;"NO": BEEP .1,1
0: BEEP .3,15: RETURN
2790 IF GUIDE=1 THEN PRINT VAL C
$-VAL B$: GO SUB 40: RETURN
2800 IF GUIDE=2 THEN PRINT VAL C
$-VAL A$: GO SUB 40: RETURN
2810 PRINT INK 6;VAL A$+VAL B$:
GO SUB 40: RETURN
2820 REM * TIMES *
2830 IF GUIDE>0 THEN GO TO 2860
2840 IF VAL A$*VAL B$=VAL C$ THE
N PRINT INK 6;"YES": BEEP .1,10:
 BEEP .3,15: RETURN
2850 PRINT INK 6;"NO": BEEP .1,1
0: BEEP .3,15: RETURN
2860 IF GUIDE=1 THEN PRINT VAL C
$/VAL B$: GO SUB 40: RETURN
2870 IF GUIDE=2 THEN PRINT VAL C
$/VAL A$: GO SUB 40: RETURN
2880 PRINT INK 6;VAL A$*VAL B$:
GO SUB 40: RETURN
2890 REM *********LESS*********
2900 LET NF=0
2910 IF CODE J$<58 THEN LET NF=1
: REM NUMBERS
2920 LET COUNT=0
2930 LET K=0
2940 LET K=K+1
2950 IF J$(K)=" " THEN LET COUNT
=COUNT+1
2960 IF COUNT=2 THEN GO TO 3000
2970 IF K<LEN J$ THEN GO TO 2940
2980 PRINT INK 5;TAB 7;"COMPARIS
ON ERROR"
2990 RETURN
3000 LET B$=J$(K+1 TO )
3010 LET A$=J$( TO K-6)
3020 IF NF=1 THEN GO TO 3050
```

```
3030 IF A$<B$ THEN PRINT INK 6;"
YES": RETURN
3040 PRINT INK 6;"NO": BEEP .1,1
0: BEEP .3,15: RETURN
3050 REM * NUMBERS *
3060 IF VAL A$<VAL B$ THEN PRINT
 INK 6;"YES": BEEP .1,10: BEEP .
3,15: RETURN
3070 PRINT INK 6;"NO": BEEP .1,1
0: BEEP .3,15: RETURN
3080 REM *********INT*********
3090 IF J$(LEN J$-1 TO )="X " TH
EN GO TO 3190
3100 LET K=0
3110 LET K=K+1
3120 IF J$(K)=" " THEN GO TO 316
0
3130 IF K<LEN J$ THEN GO TO 3110
3140 PRINT INK 5;TAB 7;"ARITHMET
IC ERROR"
3150 RETURN
3160 LET A=VAL (J$( TO K-1))
3170 IF INT A=A THEN PRINT INK 6
;"YES": BEEP .1,10: BEEP .3,15:
RETURN
3180 PRINT INK 6;"NO": BEEP .1,1
0: BEEP .3,15: RETURN
3190 LET K=0
3200 LET K=K+1
3210 IF J$(K)=" " THEN GO TO 324
0
3220 IF K<LEN J$ THEN GO TO 3200
3230 PRINT INK 5;TAB 7;"ARITHMET
IC ERROR": RETURN
3240 PRINT INK 6;INT (VAL (J$( T
O K-1)))
3250 RETURN
3260 REM *********************
3270 REM        INITIALISE
3280 CLS
3290 DIM Z$(500,30): DIM R$(200,
30)
3300 RETURN
8500 LET T=1
8505 IF A$(T)="@" THEN GO TO 853
0
8510 LET D$=A$( TO T)
8520 LET T=T+1: GO TO 8505
8530 LET A$=D$: RETURN
```

```
8600 LET B$=R$(R1): LET T=1
8605 IF B$(T)="@" THEN GO TO 863
0
8610 LET W$=B$( TO T)
8620 LET T=T+1: GO TO 8605
8630 RETURN
8700 LET B$=R$(R2): LET T=1
8705 IF B$(T)="@" THEN GO TO 873
0
8710 LET V$=B$( TO T)
8720 LET T=T+1: GO TO 8705
8730 RETURN
```

SSLISP

```
  10 REM S.S. LISP
  20 GO TO 3450: REM INITIALISAT
ION
  30 REM ***********************
  40 LET A$=A$(E TO LEN A$-1)
  50 RETURN
  60 REM ***********************
  70 PRINT
  80 LET NN=0
  90 INPUT LINE A$: PRINT ": ";A
$
 100 IF A$="" THEN BEEP 1,-15: S
TOP : REM JUST PRESS <RETURN> TO
 END RUN
 110 REM ***********************
 120 FOR J=1 TO 12
 130 LET X(J)=0: LET Y(J)=0: LET
 Z(J)=0
 140 NEXT J
 150 REM ***********************
 160 LET R=0: LET S=0: LET T=0:
LET CFIRST=0: LET CSECND=0: LET
EDGE=0
 170 FOR J=1 TO LEN A$
 180 LET B$=A$(J)
 190 IF B$="(" THEN LET S=S+1: L
ET Z(S)=J: IF T=0 THEN LET CFIRS
T=J
 200 IF B$=")" THEN LET T=T+1: L
ET Y(T)=J: IF CSECND<>0 AND EDGE
```

```
=0 THEN LET EDGE=J-1
 210 IF T=1 AND B$=")" THEN LET
CSECND=J
 220 IF B$=" " THEN LET R=R+1: L
ET X(R)=J
 230 NEXT J
 240 IF S=T THEN GO TO 300: REM
( ) BALANCE
 250 IF S<T THEN PRINT " -> MISS
ING  ("
 260 IF S>T THEN PRINT " -> MISS
ING  )"
 270 INPUT "   +   "; LINE B$
 280 LET A$=A$+B$
 290 GO TO 120
 300 IF NWDS=0 OR NN=1 THEN GO T
O 370
 310 LET M$=A$( TO X(1)-1)
 320 FOR J=1 TO NWDS
 330 IF M$=N$(J) THEN LET A$=O$(
J)+A$(LEN N$(J)+1 TO )
 340 NEXT J
 350 LET NN=1
 360 GO TO 120
 370 LET FLAG=0: LET B$="NIL"
 380 IF A$( TO 4)="EQ (" THEN LE
T FLAG=5: GO SUB 1200: GO TO 760
 390 IF A$( TO 5)="CAR (" THEN L
ET FLAG=1: GO SUB 840: GO TO 760
 400 IF A$( TO 5)="CDR (" THEN L
ET FLAG=2: GO SUB 890: GO TO 760
 410 IF A$( TO 5)="MEMQ " THEN L
ET FLAG=8: GO SUB 1510: GO TO 76
0
 420 IF A$( TO 5)="MAX (" THEN L
ET FLAG=19: GO SUB 2600: GO TO 7
60
 430 IF A$( TO 5)="MIN (" THEN L
ET FLAG=20: GO SUB 2790: GO TO 7
60
 440 IF A$( TO 6)="CONS (" THEN
LET FLAG=3: GO SUB 970: GO TO 76
0
 450 IF A$( TO 6)="ATOM (" THEN
LET FLAG=4: GO SUB 1110: GO TO 7
60
 460 IF A$( TO 6)="NULL (" THEN
LET FLAG=6: GO SUB 1330: GO TO 7
60
```

```
 470 IF A$( TO 6)="LIST (" THEN
LET FLAG=12: GO SUB 2130: GO TO
760
 480 IF A$( TO 6)="ADD1 (" THEN
LET FLAG=14: GO SUB 2180: GO TO
760
 490 IF A$( TO 6)="SUB1 (" THEN
LET FLAG=15: GO SUB 2230: GO TO
760
 500 IF A$( TO 6)="EXPT (" THEN
LET FLAG=18: GO SUB 2560: GO TO
760
 510 IF A$( TO 6)="PLUS (" THEN
LET FLAG=21: GO SUB 2820: GO TO
760
 520 IF A$( TO 6)="ONEP (" THEN
LET FLAG=31: GO SUB 2280: GO TO
760
 530 IF A$( TO 7)="ZEROP (" THEN
 LET FLAG=16: GO SUB 2280: GO TO
 760
 540 IF A$( TO 7)="EQUAL (" THEN
 LET FLAG=11: GO SUB 2000: GO TO
 760
 550 IF A$( TO 7)="MINUS (" THEN
 LET FLAG=22: GO SUB 2870: GO TO
 760
 560 IF A$( TO 7)="RECIP (" THEN
 LET FLAG=24: GO SUB 2960: GO TO
 760
 570 IF A$( TO 7)="TIMES (" THEN
 LET FLAG=26: GO SUB 3070: GO TO
 760
 580 IF A$( TO 7)="LESSP (" THEN
 LET FLAG=28: GO SUB 3160: GO TO
 760
 590 IF A$( TO 8)="DEFINE (" THE
N LET FLAG=13: GO SUB 3300: GO T
O 760
 600 IF A$( TO 8)="APPEND (" THE
N LET FLAG=9: GO SUB 1700: GO TO
 760
 610 IF A$( TO 8)="MEMBER (" THE
N LET FLAG=7: GO SUB 1380: GO TO
 760
 620 IF A$( TO 8)="MINUSP (" THE
N LET FLAG=29: GO SUB 3200: GO T
O 760
```

```
 630 IF A$( TO 9)="REVERSE (" TH
EN LET FLAG=10: GO SUB 1760: GO
TO 760
 640 IF A$( TO 9)="NUMBERP (" TH
EN LET FLAG=30: GO SUB 3250: GO
TO 760
 650 IF A$( TO 10)="QUOTIENT ("
THEN LET FLAG=23: GO SUB 2920: G
O TO 760
 660 IF A$( TO 10)="GREATERP ("
THEN LET FLAG=27: GO SUB 3120: G
O TO 760
 670 IF A$( TO 11)="REMAINDER ("
 THEN LET FLAG=25: GO SUB 3030:
GO TO 760
 680 IF A$( TO 12)="DIFFERENCE (
" THEN LET FLAG=17: GO SUB 2350
 760 IF FLAG<>0 THEN GO SUB 780
 770 GO TO 70
 780 REM ** RETURN ANSWER **
 790 PRINT "   VALUE IS..."
 800 IF B$<>"()" THEN PRINT "
";B$
 810 IF B$="()" THEN PRINT "    N
IL"
 820 RETURN
 830 REM ***********************
 840 REM         **  CAR  **
 850 IF S=2 THEN LET B$=A$(Z(2)+
1 TO X(2))
 860 IF S>2 THEN LET B$=A$(CFIRS
T TO CSECND)
 870 RETURN
 880 REM ***********************
 890 REM         **  CDR  **
 900 GO SUB 840
 910 LET LB=LEN B$+7
 920 LET B$="("+A$(LB TO EDGE+1)
 930 IF B$(LEN B$-1 TO LEN B$)="
))" THEN LET B$=B$( TO LEN B$-1)
 940 IF B$(2)=" " THEN LET B$="(
"+B$(3 TO )
 950 RETURN
 960 REM ***********************
 970 REM        **  CONS  **
 980 LET B$=A$(7 TO LEN A$-1)
 990 LET J=0
1000 IF B$(1)="(" THEN LET J=1
```

```
1010 LET J=J+1
1020 IF B$(J)="(" THEN GO TO 105
0
1030 IF J<LEN B$ THEN GO TO 1010
1040 LET B$="  * CONS ERROR *":
RETURN
1050 LET LB=LEN B$-1
1060 LET B$="("+B$( TO J-1)+B$(J
+1 TO )
1070 LET B$=B$( TO LB+1)
1080 IF B$(LEN B$-1 TO )=" )" TH
EN LET B$=B$( TO LEN B$-2)+")"
1090 RETURN
1100 REM ***********************
1110 REM       **  ATOM  **
1120 LET A$=A$(7 TO LEN A$-1)
1130 LET J=0
1140 LET J=J+1
1150 IF A$(J)=" " OR A$(J)="(" T
HEN RETURN
1160 IF J<LEN A$ THEN GO TO 1140
1170 LET B$="T"
1180 RETURN
1190 REM ***********************
1200 REM       **  EQ  **
1210 LET E=5: GO SUB 40
1220 LET J=0
1230 LET J=J+1
1240 IF A$(J)=")" THEN RETURN
1250 IF A$(J)=" " THEN GO TO 128
0
1260 IF J<LEN A$ THEN GO TO 1230
1270 RETURN
1280 LET C$=A$( TO J-1)
1290 LET A$=A$(J+1 TO )
1300 IF C$=A$ THEN LET B$="T"
1310 RETURN
1320 REM ***********************
1330 REM       **  NULL  **
1340 IF A$="NULL ()" THEN LET B$
=" * ILLEGAL - NULL NEEDS ARGUME
NT *"
1350 IF A$="NULL (())" THEN LET
B$="T"
1360 RETURN
1370 REM ***********************
1380 REM      **  MEMBER  **
1390 LET C$=A$(9 TO )
1400 LET J=1
```

```
1410 LET J=J+1
1420 IF C$(J)=")" OR C$(J)="(" T
HEN LET D$=C$( TO J): GO TO 1450
1430 IF J<LEN C$ THEN GO TO 1410
1440 RETURN
1450 LET J=LEN D$
1460 LET J=J+1
1470 IF C$(J TO J+LEN D$-1)=D$ T
HEN LET C$=C$( TO LEN C$-1): GO
TO 1630
1480 IF J<LEN C$-LEN D$ THEN GO
TO 1460
1490 RETURN
1500 REM ***********************
1510 REM       **  MEMO   **
1520 LET C$=A$(7 TO )
1530 LET J=0
1540 LET J=J+1
1550 IF C$(J)=" " THEN GO TO 158
0
1560 IF J<LEN A$ THEN GO TO 1540
1570 RETURN
1580 LET D$=C$( TO J)
1590 LET C$=C$(J+2 TO )
1600 LET C$=C$( TO LEN C$-2)+" "
1610 LET J=0
1620 LET J=J+1
1630 IF C$(J TO J+LEN D$-1)=D$ T
HEN LET B$="("+C$(J TO ): GO TO
1660
1640 IF J<LEN C$-LEN D$ THEN GO
TO 1620
1650 RETURN
1660 LET B$=B$( TO LEN B$-1)+")"
1670 IF B$(LEN B$-2 TO )=")))" T
HEN LET B$=B$( TO LEN B$-1): GO
TO 1670
1680 RETURN
1690 REM ***********************
1700 REM      **  APPEND   **
1710 LET B$=A$(9 TO )
1720 LET B$=B$( TO Y(1)-9)+" "+B
$(Z(3)-7 TO )
1730 LET B$=B$( TO LEN B$-1)
1740 RETURN
1750 REM ***********************
1760 REM      **  REVERSE   **
1770 LET B$=""
1780 LET A$=A$(11 TO ): LET A$=A
$( TO LEN A$-2)
```

```
1790 LET CT=0
1800 LET J=0
1810 LET J=J+1: IF J>LEN A$ THEN
 GO TO 1920
1820 IF A$(J)=" " THEN GO TO 185
0
1830 IF A$(J)="(" THEN GO TO 186
0
1840 GO TO 1810
1850 LET CT=CT+1: LET G$(CT)=A$(
 TO J-1): LET A$=A$(J+1 TO ): GO
 TO 1800
1860 LET J=J+1: IF A$(J TO J+1)=
"))" THEN GO TO 1980
1870 IF A$(J)=")" THEN GO TO 191
0
1880 IF J=LEN A$ THEN GO TO 1900
1890 GO TO 1860
1900 LET CT=CT+1: LET G$(CT)=A$+
")": GO TO 1930
1910 LET CT=CT+1: LET G$(CT)=A$(
 TO J): GO TO 1800
1920 LET CT=CT+1: LET G$(CT)=A$
1930 FOR M=CT TO 1 STEP -1
1940 LET B$=B$+G$(M): IF M>1 THE
N LET B$=B$+" "
1950 NEXT M
1960 LET B$="("+B$+")"
1970 RETURN
1980 LET CT=CT+1: LET G$(CT)=A$(
 TO J+1): LET A$=A$(J+2 TO ): GO
 TO 1800
1990 REM ***********************
2000 REM       **  EQUAL  **
2010 LET E=8: GO SUB 40
2020 LET M=CODE A$: IF M>47 AND
M<58 THEN GO TO 2370
2030 LET J=0
2040 LET J=J+1
2050 IF A$(J TO J+1)=") " THEN L
ET J=J+1: GO TO 1280
2060 IF A$(J TO J+2)=")))" THEN
GO TO 2100
2070 IF A$(J TO J+1)="))" THEN G
O TO 2110
2080 IF J<LEN A$ THEN GO TO 2040
2090 RETURN
2100 LET C$=A$( TO J+2): LET A$=
A$(J+4 TO ): GO TO 1300
```

```
2110 LET C$=A$( TO J+1): LET A$=
A$(J+3 TO ): GO TO 1300
2120 REM ***********************
2130 REM       **  LIST   **
2140 LET E=7: GO SUB 40
2150 LET B$="("+A$+")"
2160 RETURN
2170 REM ***********************
2180 REM       **  ADD1   **
2190 LET E=7: GO SUB 40
2200 LET B$=STR$ (VAL A$+1)
2210 RETURN
2220 REM ***********************
2230 REM       **  SUB1   **
2240 LET E=7: GO SUB 40
2250 LET B$=STR$ (VAL A$-1)
2260 RETURN
2270 REM ***********************
2280 REM       **  ZEROP **
2290 IF FLAG=16 THEN LET E=8
2300 IF FLAG=31 THEN LET E=7
2310 GO SUB 40
2320 IF A$="0" AND FLAG=16 OR A$
="1" AND FLAG=31 THEN LET B$="T"
2330 RETURN
2340 REM ***********************
2350 REM  **  TWO ARGUMENTS **
2360 LET E=13: GO SUB 40
2370 LET J=0
2380 LET J=J+1
2390 IF A$(J)=" " THEN GO TO 242
0
2400 IF J<LEN A$ THEN GO TO 2380
2410 LET B$="* ERROR - ONLY ONE
ARGUMENT *": RETURN
2420 LET P=VAL A$( TO J-1)
2430 LET Q=VAL A$(J+1 TO )
2440 IF FLAG=17 THEN LET B$=STR$
 (P-Q): RETURN
2450 IF FLAG=23 OR FLAG=25 THEN
LET B=P/Q
2460 IF FLAG=25 THEN LET B=(INT
(.5+Q*(B-INT B)*1000))/1000
2470 IF FLAG=18 THEN LET B=P↑Q
2480 IF FLAG=11 AND P=Q THEN LET
 B$="T"
2490 IF FLAG=27 AND P>Q THEN LET
 B$="T"
2500 IF FLAG=28 AND P<Q THEN LET
 B$="T"
```

```
2510 IF FLAG=32 THEN LET B=P-Q
2520 IF FLAG=11 OR FLAG>26 THEN
RETURN
2530 LET B$=STR$ (B)
2540 RETURN
2550 REM **********************
2560 REM       **  EXPT  **
2570 LET E=7: GO SUB 40
2580 GO TO 2370
2590 REM **********************
2600 REM          **  MAX  **
         ** (MIN PLUS TIMES)  **
2610 LET F$=A$( TO 3): LET A$=A$
(6 TO )
2620 LET CT=0: LET FLAG=0
2630 IF F$="TIMES" THEN LET CT=1
2640 LET J=0
2650 LET J=J+1
2660 IF A$(J)=" " THEN GO TO 269
0
2670 IF J<LEN A$ THEN GO TO 2650
2680 IF J=LEN A$ THEN LET FLAG=1
2690 LET P=VAL A$( TO J-1): IF F
LAG=0 THEN LET A$=A$(J+1 TO )
2700 IF F$<>"PLUS" AND CT=0 THEN
 LET CT=P
2710 IF F$="MAX" AND P>CT THEN L
ET CT=P
2720 IF F$="MIN" AND P<CT THEN L
ET CT=P
2730 IF F$="PLUS" THEN LET CT=CT
+P
2740 IF F$="TIMES" THEN LET CT=C
T*P
2750 IF FLAG=0 THEN GO TO 2640
2760 LET B$=STR$ (CT)
2770 RETURN
2780 REM **********************
2790 REM          **  MIN  **
2800 GO TO 2610
2810 REM **********************
2820 REM          **  PLUS  **
2830 LET F$="PLUS"
2840 LET A$=A$(7 TO )
2850 GO TO 2620
2860 REM **********************
2870 REM        **  MINUS  **
2880 LET E=8: GO SUB 40
2890 LET B$=STR$ -VAL A$
```

```
2900 RETURN
2910 REM ***********************
2920 REM    **  QUOTIENT  **
2930 LET E=11: GO SUB 40
2940 GO TO 2370
2950 REM ***********************
2960 REM      **  RECIP   **
2970 LET E=8: GO SUB 40
2980 IF A$="0" THEN LET B$=" DIV
ISION BY ZERO ILLEGAL": RETURN
2990 LET B=1/VAL A$
3000 LET B$=STR$ B
3010 RETURN
3020 REM ***********************
3030 REM    **  REMAINDER   **
3040 LET E=12: GO SUB 40
3050 GO TO 2370
3060 REM ***********************
3070 REM      **  TIMES   **
3080 LET F$="TIMES"
3090 LET A$=A$(8 TO )
3100 GO TO 2620
3110 REM ***********************
3120 REM    **  GREATERP   **
3130 LET E=11: GO SUB 40
3140 GO TO 2370
3150 REM ***********************
3160 REM      **  LESSP   **
3170 LET E=8: GO SUB 40
3180 GO TO 2370
3190 REM ***********************
3200 REM     **  MINUSP   **
3210 LET E=9: GO SUB 40
3220 IF VAL A$<0 THEN LET B$="T"
3230 RETURN
3240 REM ***********************
3250 REM     **  NUMBERP   **
3260 LET A$=A$(10 TO )
3270 IF CODE A$>44 AND CODE A$<5
8 THEN LET B$="T"
3280 RETURN
3290 REM ***********************
3300 REM     **  DEFINE   **
3310 LET A$=A$(9 TO )
3320 LET F$=A$( TO X(2)-9)
3330 LET G$=A$(X(4)-6 TO )
3340 LET J=0
3350 LET J=J+1
```

```
3360 IF G$(J)=" " THEN GO TO 339
0
3370 IF J<LEN G$ THEN GO TO 3350
3380 LET B$=" DEFINE ERROR": RET
URN
3390 LET G$=G$( TO J-1)
3400 LET NWDS=NWDS+1
3410 LET O$(NWDS)=G$: LET N$(NWD
S)=F$
3420 LET B$=F$
3430 RETURN
3440 REM ***********************
3450 REM       INITIALISE
3452 POKE 23658,8
3454 POKE 23609,40
3456 POKE 23692,255
3460 BORDER 1: PAPER 1: INK 7: C
LS
3470 DIM G$(20,20): DIM O$(20,10
): DIM N$(20,10): DIM X(12): DIM
 Y(12): DIM Z(12)
3480 LET NWDS=0: REM COUNT OF US
ER-DEFINED NEW WORDS
3490 GO TO 70
```

BBC Micro

THE AUTO MECHANIC

```
   10 REM THE AUTO MECHANIC
   20 MODE 6:VDU 19,0,4;0;
   30 PRINT"SO YOU'RE HAVING AUTOMOBILE
PROBLEMS."
   40 PRINT"LETS SEE IF WE CAN PIN DOWN
THE TROUBLE...!"
   50 PRINT
   60 PRINT"ENTER A LETTER WHICH DESCRIB
ES PROBLEM:"
   70 PRINT
   80 PRINT"  A - ENGINE WON'T TURN OVER
"
   90 PRINT"  B - ENGINE TURNS, BUT WON'
T START"
  100 PRINT"  C - STARTS, THEN STALLS ST
RAIGHT AWAY"
  110 PRINT"  D - CAR RUNS BUT STALLS A
LOT"
  120 PRINT"  E - CAR RUNS BUT IDLES ROU
GHLY"
  130 IF INKEY$(0)<>"" THEN 130
  140 A$=INKEY$(0)
  150 IF A$<"A" OR A$>"E" THEN 140
  160 GOSUB 1570
  170 ON ASC(A$)-64 GOTO 200,580,1200,12
30,1350
  180 END
  190 REM ******************************
  200 REM WON'T TURN OVER
```

```
210 PRINT"WE'LL START BY CHECKING THE
BATTERY."
220 PRINT"TURN ON THE LIGHTS."
230 PRINTTAB(6);"ARE THEY DIM?"
240 GOSUB 1520
250 IF A$="N" THEN 350:REM BATTERY O.K
260 PRINT"ARE BATTERY CABLES LOOSE OR
CORRODED?"
270 GOSUB 1520
280 IF A$="Y" THEN PRINT"TIGHTEN AND C
LEAN"
290 GOSUB 1590
300 PRINT"IS FAN BELT LOOSE?"
310 GOSUB 1520
320 IF A$="Y" THEN PRINTTAB(8);"TIGHTE
N THE FAN BELT"
330 GOSUB 1590
340 PRINT"JUMPER LEADS OR A PUSH SHOUL
D BE ENOUGH TO START THE CAR":END
350 PRINT"IS THERE LOOSENESS OR CORROS
ION AT THE "
360 PRINT"STARTER END OF THE BATTERY C
ABLE?"
370 GOSUB 1520
380 IF A$="Y" THEN PRINT"TIGHTEN AND C
LEAN THE CONNECTIONS"
390 GOSUB 1590
400 PRINT"PUT A PIECE OF METAL ACROSS
THE"
410 PRINT"SOLENOID TERMINALS. DOES STA
RTER WORK?"
420 GOSUB 1520
430 IF A$="Y" THEN PRINT"THE IGNITION
SWITCH IS PROBABLY FAULTY"
440 GOSUB 1590
```

```
  450 IF A$="Y" THEN PRINTTAB(4);"IT SHO
ULD BE REPLACED":END
  460 PRINT"DOES STARTER CLICK?"
  470 GOSUB 1520
  480 IF A$="Y" THEN PRINT"STARTER MAY B
E JAMMED":GOTO 520
  490 PRINT"AS NOTHING HAPPENED, THE SOL
ENOID IS"
  500 PRINT"PROBABLY FAULTY. A PUSH MAY
START"
  510 PRINTTAB(12);"YOUR CAR.":END
  520 REM STARTER JAMMED
  530 PRINT"TURN THE IGNITION OFF, AND P
UT THE CAR"
  540 PRINT"IN A HIGH GEAR. PUSH CAR FOR
 A FOOT OR"
  550 PRINT"SO TO POP STARTER LOOSE.":EN
D
  560 RETURN
  570 REM *****************************
  580 REM TURN OVER, WON'T START
  590 PRINT"CHECK WIRES AT POINTS FOR DA
MPNESS."
  600 PRINT"IS THERE A CHANCE AREAS ARE
DAMP?"
  610 GOSUB 1520
  620 IF A$="Y" THEN PRINT"SPRAY WITH MO
ISTURE REMOVAL SPRAY":GOSUB 1590
  630 PRINT"IS THERE DUST VISIBLE ON THE
 INSULATING"
  640 PRINTTAB(6);"PART OF THE COIL, OR
ON THE"
  650 PRINTTAB(11);"DISTRIBUTOR CAP?"
  660 GOSUB 1520
```

```
670 IF A$="Y" THEN PRINT"WIPE COIL SEC
TION AS WELL AS INSIDE"
680 IF A$="Y" THEN PRINTTAB(6);"AND OU
TSIDE OF CAP.":GOSUB 1590
690 PRINT"ENSURE ALL WIRES ARE TIGHT A
ND DRY        BEFORE CONTINUING"
700 GOSUB 1590
710 PRINT"IF CAR STILL DOES NOT START,
 IT IS TIME TO CHECK THE SPARK PLUGS:"
720 GOSUB 1590
730PRINT"DOES SPARK FROM THE END OF TH
E PLUG WIREJUMP 3/8 OF AN INCH OR MORE?"
740 GOSUB 1520
750 IF A$="Y" THEN PRINT"THE PLUGS ARE
 FAULTY":GOSUB 1590
760 IF A$="N" THEN 830
770 PRINT"ARE THE PLUGS GREASY?"
780 GOSUB 1520
790 IF A$="Y" THEN PRINT"AN EMERGENCY
REPAIR CANNOT BE MADE"
800 IF A$="Y" THEN PRINT"WHILE PLUGS A
RE IN THAT STATE.":GOSUB 1590:GOTO 770
810 IF A$="N" THEN PRINT"IF THIS IS AN
 EMERGENCY THEN TRY CLOSING"
820 IF A$="N" THEN PRINT"THE GAP TO AB
OUT HALF NORMAL":GOSUB 1590:GOTO 910
830 PRINT"IS THERE ANY SPARK AT ALL?"
840 GOSUB 1520
850 IF A$="Y" THEN 770
860 GOSUB 870:END
870 PRINT"CHECK ROTOR, COIL AND DISTRI
BUTOR CAP"
880 PRINT"FOR CRACKS. IF THERE AREN'T
ANY THERE,"
890 PRINT"IT LOOKS AS IF THE POINTS OR
 CONDENSER  IS YOUR PROBLEM"
```

```
  900 PRINT"A REPAIR MAY WELL BE NEEDED"
:RETURN
  910 PRINT"DO YOU HAVE GAS IN THE TANK?
"
  920 GOSUB 1520
  930 IF A$="N" THEN PRINT"FILL TANK AND
 TRY AGAIN":GOSUB 1590
  940 PRINT"ARE ALL FUEL AND VACUUM LINE
S SECURE?"
  950 GOSUB 1520
  960 IF A$="N" THEN PRINT"ATTEND TO THE
SE AND TRY AGAIN":GOSUB 1590
  970 PRINT"   REMOVE THE AIR CLEANER FR
OM THE"
  980 PRINTTAB(13);"CARBURETOR":GOSUB 15
90
  990 PRINT"DOES IT LOOK DRY?"
 1000 GOSUB 1520
 1010 IF A$="N" THEN 1080
 1020 PRINT"TURN THE ENGINE OVER A FEW T
IMES WITH"
 1030 PRINT"YOUR HAND SEALING THE AIR IN
TAKE":GOSUB 1590
 1040 PRINT"IS YOUR HAND WET WITH GAS?"
 1050 GOSUB 1520
 1060 IF A$="N" THEN PRINT"UNSCREW GAS C
AP IN CAS AIR VENT IS PLUGGED"
 1070 IF A$="N" THEN PRINT"THE FUEL PUMP
 MAY NOT BE WORKING":GOSUB 1080
 1080 PRINT "HAVE YOU BEEN CRANKING THE
STARTER A"
 1090 PRINT"LOT IN THE PAST FEW MINUTES?
"
 1100 GOSUB 1520
 1110 IF A$="N" THEN 1150
```

```
1120 PRINT"WAIT FOR A MINUTE OR SO, THE
N HOLD THE"
1130 PRINT"GAS PEDAL STEADILY ON THE FL
OOR WITHOUT"
1140 PRINT"PUMPING IT. THIS SHOULD GET
YOU GOING":END
1150 PRINT"THE ENGINE MAY WELL BE FLOOD
ED AT THIS"
1160 PRINT"POINT AND THE FLOOD VALVE ST
UCK OPEN.":GOSUB 1590
1170 PRINT"TAP THE SIDE OF THE CARBURET
OR"
1180 PRINT"THEN TRY THE STARTING PROCES
S AGAIN":END
1190 REM ******************************
1200 REM STARTS, THEN STALLS
1210 PRINT"THIS SUGGESTS A FAULTY BALAS
T RESISTOR  WHICH SHOULD BE REPLACED":EN
D
1220 REM ******************************
1230 REM RUNS, STALLS A LOT
1240 PRINT"THE PROBLEM IS EITHER CAUSED
 BY :"
1250 PRINTTAB(7);"SHORTING (OR LOOSE) W
IRES ;"
1260 PRINTTAB(7);"A WEAK SPARK; OR"
1270 PRINTTAB(7);"A FAULT IN THE FUEL S
YSTEM"
1280 PRINT"CHECK FIRST FOR LOOSE OR SHO
RTING WIRES":GOSUB 1590
1290 PRINT"IF THEY ARE NOT OK, REPAIR.
IF THEY ARE,IT COULD BE THE SPARK PLUGS"
1300 GOSUB 870
1310 PRINT"THERE IS A FINAL CHECK WE CA
N TRY ON"
```

```
1320 PRINT"YOUR SPARK PLUGS":GOSUB 1590
1330 GOTO 1360
1340 REM ******************************
1350 REM RUNS, ROUGH IDLE
1360 PRINT"IT COULD WELL BE THAT ONE OR
MORE OF"
1370 PRINT"YOUR SPARK PLUGS ARE FAULTY.
":GOSUB 1590
1380 PRINT"DISCONNECT THEM ONE AT A TIM
E. THE ONES"
1390 PRINT"WHICH DO NOT CAUSE THE ENGIN
E IDLE TO"
1400 PRINT"DROP ARE FAULTY. CHECK THESE
 THEN"
1410 PRINT"RETURN TO THE SYSTEM.":GOSUB
 1590
1420 PRINT"DID TEST SHOW ANY PLUGS WERE
 FAULTY?"
1430 GOSUB 1520
1440 IF A$="Y" THEN PRINT"REPLACE ALL P
LUGS IF YOU CAN, OR JUST"
1450 IF A$="Y" THEN PRINT"THE ONES WHIC
H TESTED FAULTY":GOSUB 1590
1460 PRINT"THE MOST COMMON CAUSE OF A B
AD IDLE,"
1470 PRINT"ASSUMING THAT THE PLUGS ARE
OK, IS THAT"
1480 PRINT"YOUR GAS MIXTURE IS SET TO R
ICH"
1490 PRINT"SO YOU SHOULD ADJUST THIS":E
ND
1500 PRINT"ADJUST THE IDLE SPEED-SCREW
ON THE         THROTTLE LINKAGE":END
1510 REM ******************************
1520 PRINTTAB(16);"(Y - YES, N - NO)?"
```

```
1530 IF INKEY$(0)<>"" THEN 1530
1540 A$=INKEY$(0)
1550 SOUND 16,-RND(15),RND(256)-1,5
1560 IF A$<>"Y" AND A$<>"N" THEN 1540
1570 PRINTTAB(22);"> OK    ";A$;" <"
1580 VDU 7
1590 FOR T=1 TO 2000:NEXT T
1600 PRINT
1610 RETURN
```

MEDICI

```
   10 REM MEDICI - PERSONAL CHECKUP
   20 MODE 6:VDU 19,0,4;0;
   30 GOSUB 1250
   40 PRINT "A - I AM BADLY OVERWEIGHT"
   50 PRINT "B - I AM FAIRLY OVERWEIGHT"
   60 PRINT "C - I AM SLIGHTLY OVERWEIGH
T"
   70 PRINT "D - MY WEIGHT IS ABOUT RIGH
T"
   80 PRINT "E - I AM THINNER THAN I SHO
ULD BE"
   90 GOSUB 1170
  100 WEIGHT=5*(ASC(A$)-68):IF A$="E" TH
EN WEIGHT=0
  110 PRINT WEIGHT
  120 GOSUB 1250
  130 PRINT "I ENGAGE IN EXERCISE, THAT"
  140 PRINT "RAISES MY HEARTBEAT TO 120
OR MORE,"
  150 PRINT "FOR AT LEAST THE FOLLOWING
NUMBER"
  160 PRINT TAB(8);"OF HOURS A WEEK:"
  170 PRINT
```

```
  180 PRINT "A - LESS THAN A QUARTER"
  190 PRINT "B - MORE THAN A QUARTER, UP
 TO                THREE QUARTERS"
  200 PRINT "C - FROM THREE-QUARTERS OF
AN HOUR            UP TO ONE AND A HALF"
  210 PRINT "D - FROM ONE AND A HALF TO
                   TWO AND A HALF"
  220 PRINT "E - MORE THAN TWO AND A HAL
F HOURS"
  230 GOSUB 1170
  240 EXERCISE=5*(ASC(A$)-63)-5:IF A$="A
" THEN EXERCISE=0
  250 PRINT EXERCISE
  260 GOSUB 1250
  270 PRINT "WHEN DRIVING:":PRINT
  280 PRINT "A - I HARDLY EVER WEAR A SE
AT BELT"
  290 PRINT "B - I WEAR A SEAT BELT AROU
ND A QUARTER       OF THE TIME"
  300 PRINT "C - I WEAR A SEAT BELT EVER
Y SECOND           JOURNEY"
  310 PRINT "D - I WEAR A SEAT BELT FOR
MOST, BUT NOT      ALL TRIPS"
  320 PRINT "E - I ALWAYS WEAR A SEAT BE
LT"
  330 GOSUB 1170
  340 SEATBELT=2*(ASC(A$)-65)
  350 PRINT SEATBELT
  360 GOSUB 1250
  370 PRINT "I AM CONSCIOUS OF NUTRITION
 AND TRY        TO EAT HEALTHILY:"
  380 PRINT
  390 PRINT "A - ALL OF THE TIME"
  400 PRINT "B - NEARLY ALL OF THE TIME"
  410 PRINT "C - A FAIR PROPORTION OF TH
E TIME"
```

```
  420 PRINT "D - FROM TIME TO TIME"
  430 PRINT "E - HARDLY AT ALL"
  440 GOSUB 1170
  450 DIET=-ASC(A$)+69
  460 PRINT DIET
  470 GOSUB 1250
  480 PRINT "SMOKING (A CIGAR COUNTS AS
A CIGARETTE)"
  490 PRINT
  500 PRINT "A - NOT AT ALL"
  510 PRINT "B - LESS THAN 15 CIGARETTES
 A DAY"
  520 PRINT "C - 15 TO 25 CIGARETTES A D
AY"
  530 PRINT "D - 26 TO 42 CIGARETTES A D
AY"
  540 PRINT "E - MORE THAN 42 CIGARETTES
 A DAY"
  550 GOSUB 1170
  560 SMOKING=-7*(ASC(A$)-65)
  570 PRINT SMOKING
  580 GOSUB 1250
  590 PRINT "ALCOHOL - HOW MANY DRINKS (
ON AVERAGE) DO YOU     HAVE EACH DAY"
  600 PRINT
  610 PRINT "A - NONE"
  620 PRINT "B - LESS THAN 3"
  630 PRINT "C - 3 TO 6"
  640 PRINT "D - 7 TO 9"
  650 PRINT "E - MORE THAN 9"
  660 GOSUB 1170
  670 DRINK=-30
  680 IF A$="A" THEN DRINK=0
  690 IF A$="B" THEN DRINK=1
  700 IF A$="C" THEN DRINK=DRINK/5
```

```
710 IF A$="D" THEN DRINK=DRINK/2
720 PRINT DRINK
730 GOSUB 1250
740 PRINT "IN GENERAL, HOW STRESSFUL W
OULD YOU SAY";
750 PRINT " YOUR LIFE HAS BEEN IN THE
LAST 6 MONTHS"
760 PRINT
770 PRINT "A - EXTREMELY STRESSFUL"
780 PRINT "B - FAIRLY STRESSFUL"
790 PRINT "C - SLIGHTLY STRESSFUL"
800 PRINT "D - NEUTRAL"
810 PRINT "E - NOT STRESSFUL"
820 GOSUB 1170
830 STRESS=INT(2.5*(ASC(A$)-69))
840 PRINT STRESS
850 GOSUB 1300:CLS
860 PRINT "PERSONAL ASSESSMENT FROM ME
DICI:"
870 PRINT
880 PRINT TAB(8);"WEIGHT:";WEIGHT
890 PRINT TAB(6);"EXERCISE:";EXERCISE
900 PRINT TAB(4);"CAR SAFETY:";SEATBEL
T
910 PRINT TAB(5);"NUTRITION:";DIET
920 PRINT TAB(7);"SMOKING:";SMOKING
930 PRINT TAB(7);"ALCOHOL:";DRINK
940 PRINT TAB(8);"STRESS:";STRESS
950 GOSUB 1300
960 ANT=WEIGHT+EXERCISE+SEATBELT+DIET+
SMOKING+DRINK+STRESS
970 GOSUB 1300:PRINT
980 PRINT "     YOUR RAW RATING IS ";AN
T:PRINT
```

```
 990 PRINT "  ON A SCALE WHERE ZERO IS
AVERAGE,"
 1000 PRINT "THE LOWEST RATING IS BELOW
-80 AND"
 1010 PRINT "   THE HIGHEST IS OVER 30"
 1020 GOSUB 1300:PRINT
 1030 IF ANT<6 AND ANT>-6 THEN A$="AVERA
GE":L$="62 TO 73    72 TO 78"
 1040 IF ANT<-5 AND ANT>-21 THEN A$="BEL
OW AVERAGE":L$="60 TO 66    65 TO 71"
 1050 IF ANT<-20 THEN A$="POOR":L$="60 O
R LESS 65 OR LESS"
 1060 IF ANT<-45 THEN A$="VERY POOR"
 1070 IF ANT<-60 THEN A$="VERY, VERY POO
R"
 1080 IF ANT>5 AND ANT<15 THEN A$="GOOD"
:L$="74 TO 80    79 TO 85"
 1090 IF ANT>14 THEN A$="EXTREMELY GOOD"
:L$="81+      86+"
 1100 PRINT "THIS INDICATES YOUR HEALTH
STATUS         IS ";A$
 1110 PRINT
 1120 PRINT "LIFE EXPECTANCY:"
 1130 PRINT TAB(3);"MALE         FEMALE"
 1140 PRINT TAB(3);L$
 1150 END
 1160 REM ******************************
 1170 REM ACCEPT INPUT
 1180 IF INKEY$(0)<>"" THEN 1180
 1190 A$=INKEY$(0)
 1200 IF A$<"A" OR A$>"E" THEN 1190
 1210 PRINT:PRINT TAB(12);"OK     ";A$
 1220 RETURN
 1230 REM ******************************
 1240 REM DELAY/SPACE OUT
 1250 FOR J=1 TO 1000:NEXT J
```

```
 1260 CLS
 1270 PRINT:PRINT:PRINT:PRINT
 1280 PRINT "WHICH OF THE FOLLOWING IS C
LOSEST        TO THE TRUTH (SELECT ONE):"
 1290 PRINT
 1300 FOR J=1 TO 400:NEXT J
 1310 RETURN
```

FUZZY RITA

```
   10 REM FUZZY RITA
   20 PROCinitialise
   30 PROCout_put_options
   40 PROCdiscrimination_options
   50 PROCask_question
   60 PROCmake_decision_and_update_base
   70 PRINT"PRESS <RETURN> TO CONTINUE";
   80 IF X$<>"" THEN INPUT I$:GOTO 50
   90 PRINT" TRAINING"
  100 PRINT"OR ANY KEY THEN <RETURN> TO
USE RITA"
  110 INPUT X$:GOTO 50
  120 END
  130 REM ****************************
  140 DEF PROCask_question
  150 CLS
  160 PRINT"HERE ARE THE SUBJECTS I CAN
          DISCRIMINATE BETWEEN"
  170 PRINT
  180 FOR J=1 TO TT
  190 PRINT" >  ";A$(J)
  200 NEXT J
  210 GOSUB 1500
  220 IF X$="" THEN PRINT"THINK OF ONE,
THEN PRESS <RETURN>"
```

```
  230 IF X$<>"" THEN PRINT"I AM READY NO
W TO DETERMINE WHICH ONE    YOU HAVE"
  240 IF X$="" THEN INPUT J$
  250 ADD=.5
  260 FOR J=1 TO DQ
  270 ADD=ADD+ADD
  280 GOSUB 1500
  290 IF X$<>"" AND TT>2 THEN 390:REM CH
ECK IF QUESTION CAN BE JUMPED
  300 PRINT"ENTER A NUMBER FROM "
  310 PRINT"1 (TRUE) TO 0 (FALSE) ($ TO
END RUN)"
  320 PRINT:PRINT E$(J);
  330 INPUT H$:IF H$="$" THEN PRINT:PRIN
T"THANK YOU":PRINT:END
  340 C(J)=VAL(H$)
  350 C(J)=ADD*C(J)
  360 NEXT J
  370 ENDPROC
  380 REM ******************************
  390 REM CHECK IF QUESTION CAN BE JUMPE
D
  400 JUMP=1
  410 FOR W=1 TO TT
  420 IF ABS(B(W,J)-B(1,J))>.7 THEN JUMP
=0
  430 NEXT W
  440 IF JUMP=0 THEN 300
  450 C(JUMP)=B(W,J)
  460 GOTO 360
  470 REM ******************************
  480 DEF PROCmake_decision_and_update_b
ase
  490 FOR J=1 TO TT
  500 D(J)=0:E(J)=0:F(J)=0
  510 NEXT J
```

```
520 ADD=.5
530 FOR J=1 TO TT
540 ADD=ADD+ADD
550 FOR X=1 TO DQ
560 REM PLAY WITH VALUES IN NEXT THREE
 LINES FOR MOST EFFICIENT RESULTS
570 IF C(X)=B(J,X) THEN D(J)=D(J)+1
580 IF ABS(C(X)-B(J,X))<.6*ADD THEN E(
J)=E(J)+.4
590 IF ABS(C(X)-B(J,X))<1.2 *ADD THEN
F(J)=F(J)+.1
600 NEXT X
610 NEXT J
620 A1=1:A2=1:A3=1
630 F1=1:F2=1:F3=1
640 FOR J=1 TO TT
650 IF D(J)>F1 THEN F1=D(J):A1=J
660 IF E(J)>F2 THEN F2=E(J):A2=J
670 IF F(J)>F3 THEN F3=F(J):A3=J
680 NEXT J
690 REM ** ANNOUNCE RESULT **
700 PRINT
710 CFLG=0
720 PRINT"THE MOST LIKELY RESULT IS ";
A$(A1)
730 IF A2<>A1 THEN PRINT"THE NEXT MOST
 LIKELY IS ";A$(A2):CFLG=1
740 IF A3<>A2 AND A3<>A1 THEN PRINT"TH
E NEXT MOST LIKELY IS ";A$(A3):CFLG=2
750 PRINT
760 PRINT"IS THE MOST LIKELY RESULT CO
RRECT (Y/N)";
770 INPUT F$
780 IF F$<>"Y" AND F$<>"N" THEN 770
790 IF F$="Y" AND X$<>"" THEN ENDPROC
```

```
 800 IF F$="Y" THEN 980
 810 IF TT=2 AND A1=1 THEN A1=2:GOTO 98
0
 820 IF TT=2 THEN A1=1:GOTO 980
 830 IF CFLG=0 THEN 890
 840 PRINT"IS MY SECOND CHOICE CORRECT
(Y/N)";
 850 INPUT F$
 860 IF F$="N" THEN 890
 870 IF CFLG=1 THEN A1=A2:GOTO 980
 880 IF CFLG=2 THEN A1=A3:GOTO 980
 890 GOSUB 1500
 900 FOR J=1 TO TT
 910 PRINT J;"- ";A$(J)
 920 NEXT J
 930 PRINT
 940 PRINT"WHICH IS THE CORRECT ONE";
 950 INPUT A1
 960 IF A1<1 OR A1>TT THEN 950
 970 REM ** EDUCATING RITA **
        (UPDATE KNOWLEDGE BASE)
 980 FOR J=1 TO DQ
 990 IF B(A1,J)<>0 THEN B(A1,J)=(C(J+5*
B(A1,J))/6)
1000 IF B(A1,J)=0 THEN B(A1,J)=C(J)
1010 B(A1,J)=INT(10*B(A1,J))/10
1020 NEXT J
1030 PRINT
1040 IF U$="" THEN ENDPROC
1050 FOR J=1 TO TT
1060 PRINT:GOSUB 1500
1070 PRINT A$(J)
1080 PRINT
1090 FOR K=1 TO DQ
1100 PRINT E$(K);" ";B(J,K)
```

```
 1110 NEXT K
 1120 NEXT J
 1130 PRINT
 1140 ENDPROC
 1150 REM *****************************
 1160 DEF PROCout_put_options
 1170 TT=0
 1180 TT=TT+1
 1190 GOSUB 1500
 1200 PRINT"ENTER OUTPUT OPTION NUMBER"T
T"            (PRESS <RETURN> TO END)"
 1210 INPUT A$(TT)
 1220 IF A$(TT)="" OR TT=51 THEN TT=TT-1
:ENDPROC
 1230 GOTO 1180
 1240 REM *****************************
 1250 DEF PROCdiscrimination_options
 1260 CLS
 1270 FOR J=1 TO TT
 1280 PRINTA$(J)
 1290 NEXT J
 1300 DQ=0
 1310 DQ=DQ+1
 1320 GOSUB 1500
 1330 PRINT"   ENTER QUESTION NUMBER"DQ"
           (PRESS <RETURN> TO END)"
 1340 INPUTE$(DQ)
 1350 IF E$(DQ)="" OR DQ=51 THEN DQ=DQ-1
:ENDPROC
 1360 GOTO 1310
 1370 REM *****************************
 1380 DEF PROCinitialise
 1390 CLS
 1400 REM REDUCE ARRAYS IN THE NEXT LINE
 IN ACCORDANCE WITH YOUR NEEDS
```

```
 141Ø DIM A$(5Ø),B(5Ø,5Ø),C(5Ø),D(5Ø),E$
(5Ø),E(5Ø),F(5Ø),G(5Ø)
 142Ø X$=""
 143Ø PRINT"PRESS ANY KEY, THEN <RETURN>
 IF YOU "
 144Ø PRINT"WANT TO SEE THE UPDATED KNOW
LEDGE BASE"
 145Ø PRINT"        AFTER EACH RUN; JUST
PRESS"
 146Ø PRINT"          <RETURN> IF YOU DON
'T"
 147Ø INPUT U$
 148Ø CLS
 149Ø ENDPROC
 15ØØ PRINT"-----------------------------
------------"
 151Ø RETURN
```

SCALING

```
 1Ø REM SCALING
 2Ø DIM X(5Ø),Z(5Ø)
 3Ø PRINT"
 {CLR}"
 4Ø INPUT "HIGHEST VALUE";A
 5Ø INPUT "LOWEST VALUE";B
 6Ø A=A+.ØØ1
 7Ø B=B+.ØØ1
 8Ø C=(A-B)/5Ø
 9Ø X(Ø)=B
```

```
100 FOR J=1 TO 50
110 X(J)=X(J-1)+C
120 Z(J)=J/50
130 PRINT Z(J),X(J)
140 NEXT J
150 DFF=(X(2)-X(1))/2
160 COUNT=0
170 COUNT=COUNT+1
180 PRINT"ENTER VALUE";Q$
190 INPUT Q$
200 IF Q$="" THEN END
210 Q=VAL(Q$)
220 IF Q<B OR Q>A THEN 180
230 FOR J=1 TO 50
240 IF ABS(Q-X(J))<DFF THEN PRINTCOUNT;"
-";Z(J)
250 NEXT J
260 GOTO 170
```

HASTE

```
10 REM HASTE
20 DIM A$(255),B$(255)
30 F$="":F=0:CLS
40 REM ****************************
50 FLAG=0
60 INPUT "> "D$
70 IF D$="" THEN 60
80 IF LEFT$(D$,1)="?" THEN 200
90 E=0
```

```
100 E=E+1
110 IF MID$(D$,E,1)="*" THEN 140
120 IF E<LEN(D$) THEN 100
130 PRINT "INVALID ENTRY":GOTO 50
140 IF FLAG=3 THEN RETURN
150 F=F+1:IF F=256 THEN END
160 A$(F)=LEFT$(D$,E-1)
170 B$(F)=MID$(D$,E+1)
180 GOTO 50
190 REM **************************
200 REM INTERROGATE
210 FLAG=4:true=0
220 IF RIGHT$(D$,1)="/" THEN FLAG=3
230 FOR J=1 TO LEN(D$)-5
240 IF MID$(D$,J,5)=" AND " THEN FLAG=
5:true=J
250 NEXT J
260 IF FLAG=5 THEN 410
270 IF LEFT$(D$,3)="?/*" THEN FLAG=1
280 IF LEFT$(D$,4)="?/*/" THEN FLAG=2
290 IF FLAG=3 THEN GOSUB 90:F$=MID$(D$
,2,E-2)
300 IF FLAG=1 THEN F$=MID$(D$,4)
310 E=0
320 E=E+1
330 IF A$(E)="" AND FLAG=4 AND true=0
THEN PRINT "FALSE"
340 IF A$(E)="" THEN 520
350 IF FLAG=4 AND "?"+A$(E)+"*"+B$(E)=
D$ THEN PRINT "TRUE":true=1
360 IF FLAG=3 AND F$=A$(E) THEN PRINT
B$(E)
370 IF FLAG=2 THEN PRINT A$(E);"*";B$(
E)
380 IF FLAG=1 AND F$=B$(E) THEN PRINT
A$(E)
```

```
390 IF E<255 THEN 320
400 REM **************************
410 F$=MID$(D$,4,true-4):G$=MID$(D$,tr
ue+7)
420 E=0
430 E=E+1
440 IF A$(E)="" THEN 520
450 IF B$(E)=F$ THEN 470
460 IF E<255 THEN 430
470 H=0
480 H=H+1
490 IF B$(H)="" THEN 460
500 IF B$(H)=G$ AND A$(E)=A$(H) THEN P
RINT A$(E)
510 IF H<255 THEN 480
520 PRINT TAB(5);" > END OF ANSWER <"
530 GOTO 50
```

EASLE

```
 10 REM EASEL
 20 MODE 6:VDU 19,0,4;0;
 30 DIM X(40),Y(40),Z(40)
 40 PRINT
 50 INPUT"> "A$
 60 IF A$="" THEN END
 70 REM ******************************
**
 80 FOR J=1 TO 40
 90 X(J)=0:Y(J)=0:Z(J)=0
100 NEXT J
110 REM ******************************
**
120 R=0:S=0:T=0:CFIRST=0:CSECND=0:EDGE
=0
```

```
  130 FOR J=1 TO LEN(A$)
  140 B$=MID$(A$,J,1)
  150 IF B$="(" THEN S=S+1:Z(S)=J:IF T=0
 THEN CFIRST=J
  160 IF B$=")" THEN T=T+1:Y(T)=J:IF CSE
CND<>0 AND EDGE=0 THEN EDGE=J-1
  170 IF T=1 AND B$=")" THEN CSECND=J
  180 IF B$=" " THEN R=R+1:X(R)=J
  190 NEXT J
  200 IF S=T THEN 260:REM ( ) BALANCE
  210 IF S<T THEN PRINT" -> MISSING ("
  220 IF C>T THEN PRINT" -> MISSING )"
  230 INPUT"  >   ",B$
  240 A$=A$+B$
  250 GOTO 80
  260 FLAG=0
  270 IF LEFT$(A$,5)="CAR (" THEN FLAG=1
  280 IF LEFT$(A$,5)="CDR (" THEN FLAG=2
  290 IF LEFT$(A$,6)="CONS (" THEN FLAG=
3
  300 IF LEFT$(A$,6)="ATOM (" THEN FLAG=
4
  310 IF LEFT$(A$,4)="EQ (" THEN FLAG=5
  320 IF LEFT$(A$,6)="NULL (" THEN FLAG=
6
  330 ON FLAG GOSUB 420,470,550,690,780,
920
  340 IF FLAG<>0 THEN GOSUB 360
  350 GOTO 40
  360 REM ** RETURN ANSWER **
  370 PRINT"   VALUE IS..."
  380 IF B$<>"()" THEN PRINT"    ";B$
  390 IF B$="()" THEN PRINT"   NIL"
  400 RETURN
  410 REM ******************************
**
```

```
420 REM                   ** CAR **
430 IF S=2 THEN B$=MID$(A$,Z(2)+1,X(2)
-Z(2))
440 IF S>2 THEN B$=MID$(A$,CFIRST,CSEC
ND-CFIRST+1)
450 RETURN
460 REM ********************************
**
470 REM                   ** CDR **
480 GOSUB 420
490 LB=LEN(B$)+7
500 B$="("+MID$(A$,LB,EDGE-1)
510 IF RIGHT$(B$,2)="))" THEN B$=LEFT$
(B$,LEN(B$)-1)
520 IF MID$(B$,2,1)=" " THEN B$="("+MI
D$(B$,3)
530 RETURN
540 REM ********************************
**
550 REM                   ** CONS **
560 B$=MID$(A$,7,LEN(A$)-1)
570 J=0
580 IF LEFT$(B$,1)="(" THEN J=1
590 J=J+1
600 IF MID$(B$,J,1)="(" THEN 630
610 IF J<LEN(B$) THEN 590
620 B$="    > CONS ERROR <":RETURN
630 LB=LEN(B$)-1
640 B$="("+LEFT$(B$,J-1)+MID$(B$,J+1)
650 B$=LEFT$(B$,LB)
660 IF RIGHT$(B$,2)=" )" THEN B$=LEFT$
(B$,LEN(B$)-2)+" )"
670 RETURN
680 REM ********************************
**
```

```
690 REM                 ** ATOM **
700 A$=MID$(A$,7,LEN(A$)-1)
710 J=0:B$="NIL"
720 J=J+1
730 IF MID$(A$,J,1)=" " OR MID$(A$,J,1
)="(" THEN RETURN
740 IF J<LEN(A$) THEN 720
750 B$="T"
760 RETURN
770 REM ******************************
**
780 REM                   ** EQ **
790 A$=MID$(A$,5)
800 A$=LEFT$(A$,LEN(A$)-1)
810 J=0:B$="NIL"
820 J=J+1
830 IF MID$(A$,J,1)=")" THEN RETURN
840 IF MID$(A$,J,1)=" " THEN 870
850 IF J<LEN(A$) THEN 820
860 RETURN
870 C$=LEFT$(A$,J-1)
880 A$=MID$(A$,J+1)
890 IF A$=C$ THEN B$="T"
900 RETURN
910 REM ******************************
**
920 REM                 ** NULL **
930 B$="NIL"
940 IF A$="NULL ()" THEN PRINT" ILLEGA
L - NULL NEEDS ARGUMENT":B$=""
950 IF A$="NULL (())" THEN B$="T"
960 RETURN
```

PROLOG-A

```
   10 REM PROLOG-A (SIMPLE FRONT END)
   20 REM * ALL INPUT IN UPPER CASE *
   30 GOTO 50
   40 PRINT "NO (MORE) ANSWERS":RETURN
   50 MODE6:VDU 19,0,4;0;:PROCinitialise
   60 REM ***************************
   70 PRINT
   80 INPUT "&c."J$
   90 IF J$="" THEN END
  100 IF J$="LIST ALL" THEN GOSUB 1860:G
OTO 70
  110 IF LEFT$(J$,5)="LIST " THEN J$=MID
$(J$,5)+" ":GOSUB 990:GOTO 70
  120 IF RIGHT$(J$,1)<>")" THEN PRINT "1
.":INPUT M$:J$=J$+M$:GOTO 120
  130 LJ=LEN(J$)
  140 J$=LEFT$(J$,LJ-1)+" ":REM STRIP FI
NAL ), REPLACE WITH SPACE
  150 LJ=LEN(J$)
  160 FLAG=0
  170 IF LEFT$(J$,4)="ADD(" THEN J$=MID$
(J$,5):FLAG=1
  180 RULEFLAG=0:PLUSFLAG=0:ARITHFLAG=0
  190 FOR R=1 TO LEN(J$)
  200 IF MID$(J$,R,4)=" IF " THEN RULEFL
AG=R:FLAG=6
  210 IF MID$(J$,R,5)=" AND " THEN PLUSF
LAG=R
  220 IF MID$(J$,R,4)="SUM(" THEN ARITHF
LAG=1
  230 IF MID$(J$,R,6)="TIMES(" THEN ARIT
HFLAG=2
```

```
  24Ø IF MID$(J$,R,6)=" LESS " THEN ARIT
HFLAG=3
  25Ø IF MID$(J$,R,3)="INT" THEN ARITHFL
AG=4
  26Ø NEXT R
  27Ø IF LEFT$(J$,3)="IS(" THEN J$=MID$(
J$,4):FLAG=2
  28Ø IF LEFT$(J$,1Ø)="WHICH(X : " THEN
J$=MID$(J$,11):FLAG=3
  29Ø IF LEFT$(J$,16)="WHICH((X Y) : X "
 THEN J$=MID$(J$,17):FLAG=4
  3ØØ IF FLAG=Ø THEN PRINT "SYNTAX ERROR
":GOTO 7Ø
  31Ø LJ=LEN(J$)
  32Ø REM NOW SEND TO RELEVANT SUBROUTIN
ES
  33Ø IF PLUSFLAG<>Ø THEN GOSUB 195Ø:GOT
O 7Ø:REM ENCODE RULE CONTAINING AND
  34Ø IF RULEFLAG<>Ø AND FLAG<>5 THEN GO
SUB 111Ø:REM ENCODE RULE
  35Ø IF ARITHFLAG<>Ø THEN GOSUB 243Ø:GO
TO 7Ø:REM ARITHMETIC
  36Ø IF RIGHT$(J$,3)=" X " OR RIGHT$(J$
,3)=" Y " THEN J$=LEFT$(J$,LJ-2)+" "
  37Ø LJ=LEN(J$)
  38Ø IF FLAG=1 THEN GOSUB 44Ø:REM ADD
  39Ø IF FLAG=2 THEN GOSUB 52Ø:REM IS
  4ØØ IF FLAG=3 THEN GOSUB 61Ø:REM WHICH
  41Ø IF FLAG=4 THEN GOSUB 83Ø:REM WHICH
2
  42Ø GOTO 7Ø
  43Ø REM *****************************
  44Ø REM ADD
  45Ø K=Ø
  46Ø K=K+1
```

```
  470 IF Z$(K)="" THEN Z$(K)=J$:RETURN
  480 IF K<1000 THEN 460
  490 PRINT "MEMORY FULL"
  500 RETURN
  510 REM **************************
  520 REM IS
  530 K=0
  540 K=K+1
  550 IF Z$(K)="" THEN 580
  560 IF Z$(K)=J$ THEN PRINT "YES":GOTO
590
  570 IF K<1000 THEN 540
  580 PRINT "NO"
  590 RETURN
  600 REM **************************
  610 REM WHICH
  620 IF LEFT$(J$,1)="X" THEN 710
  630 J$=LEFT$(J$,LJ-1)
  640 K=0
  650 K=K+1
  660 IF Z$(K)="" THEN 690
  670 IF J$=LEFT$(Z$(K),LEN(J$)) THEN PR
INT RIGHT$(Z$(K),(LEN(Z$(K))-LEN(J$)))
  680 IF K<1000 THEN 650
  690 GOSUB 40
  700 RETURN
  710 REM * QUERY STARTS WITH X *
  720 J$=MID$(J$,3,LEN(J$)-3)
  730 LJ=LEN(J$)
  740 K=0
  750 K=K+1
  760 IF Z$(K)="" THEN 800
  770 Q$=MID$(Z$(K),LEN(Z$(K))-LJ,LJ)
  780 IF Q$=J$ THEN PRINT LEFT$(Z$(K),LE
N(Z$(K))-(LJ+2))
```

```
790 IF K<1000 THEN 750
800 GOSUB 40
810 RETURN
820 REM **************************
830 REM WHICH2
840 J$=LEFT$(J$,LJ-2)
850 LJ=LEN(J$)
860 K=0
870 K=K+1
880 IF Z$(K)="" THEN 960
890 LFLAG=0
900 FOR L=1 TO LEN(Z$(K))-LJ
910 IF MID$(Z$(K),L,LJ)=J$ THEN LFLAG=
L
920 NEXT L
930 IF LFLAG=0 THEN 950
940 PRINT LEFT$(Z$(K),LFLAG-2);MID$(Z$
(K),(LFLAG+LJ))
950 IF K<1000 THEN 870
960 GOSUB 40
970 RETURN
980 REM **************************
990 REM LIST
1000 K=0
1010 K=K+1
1020 IF Z$(K)="" THEN RETURN
1030 LFLAG=0
1040 FOR L=1 TO LEN(Z$(K))-LEN(J$)
1050 IF MID$(Z$(K),L,LEN(J$))=J$ THEN L
FLAG=1
1060 NEXT L
1070 IF LFLAG=1 THEN PRINT Z$(K)
1080 IF K<1000 THEN 1010
1090 RETURN
1100 REM **************************
```

```
 1110 REM FORM RULES
 1120 R=RULEFLAG
 1130 E$=LEFT$(J$,R):F$=MID$(J$,R+4)
 1140 IF LEFT$(E$,1)<>"X" THEN PRINT "RU
LE ERROR":GOTO 70
 1150 REM NEXT LINE DETECTS INPUTS LIKE
          X EATS Y IF X IS-A Y
 1160 IF RIGHT$(F$,2)="Y " THEN 1390
 1170 PRINT TAB(18);"COMPILING RULE"
 1180 FOR T=1 TO 100
 1190 R$(T)=""
 1200 NEXT T
 1210 E$=MID$(E$,3):F$=MID$(F$,3)
 1220 K=0:RR=0
 1230 K=K+1
 1240 IF Z$(K)="" THEN 1300
 1250 IF RIGHT$(Z$(K),LEN(F$))<>F$ THEN
1370
 1260 RR=RR+1
 1270 R$(RR)=LEFT$(Z$(K),(LEN(Z$(K))-LEN
(F$)))+E$
 1280 PRINT "> ";R$(RR)
 1290 GOTO 1230
 1300 IF RR=0 THEN RETURN
 1310 RC=0
 1320 RC=RC+1
 1330 Z$(K)=R$(RC)
 1340 IF K<1000 THEN K=K+1
 1350 IF RC<RR THEN 1320
 1360 RETURN
 1370 IF K<1000 THEN 1230
 1380 RETURN
 1390 REM * RULE WITH 2 VARIABLES *
 1400 FOR T=1 TO 100
 1410 R$(T)=""
 1420 NEXT T
```

```
1430 K=0:RR=0
1440 IF K<1000 THEN RETURN
1450 K=K+1
1460 IF Z$(K)="" THEN 1770
1470 REM SPLIT INTO THREE WORDS
1480 Q$=Z$(K)
1490 J=0
1500 J=J+1
1510 IF MID$(Q$,J,1)=" " THEN 1540
1520 IF J<LEN(Q$) THEN 1500
1530 PRINT "RULE COMPILING ERROR":GOTO
70
1540 A$=LEFT$(Q$,J)
1550 Q$=MID$(Q$,J+1)
1560 J=0
1570 J=J+1
1580 IF MID$(Q$,J,1)=" " THEN 1610
1590 IF J<LEN(Q$) THEN 1570
1600 PRINT "RULE COMPILING ERROR":GOTO
70
1610 B$=LEFT$(Q$,J)
1620 Q$=MID$(Q$,J+1)
1630 J=0
1640 J=J+1
1650 IF MID$(Q$,J,1)=" " THEN 1680
1660 IF J<LEN(Q$) THEN 1640
1670 PRINT "RULE COMPILING ERROR":GOTO
70
1680 PRINT TAB(18);"COMPILING RULE"
1690 C$=LEFT$(Q$,J)
1700 M$=MID$(F$,3,LEN(B$))
1710 IF B$<>M$ THEN 1440
1720 RR=RR+1
1730 N$=MID$(E$,3,LEN(E$)-4)
1740 R$(RR)=A$+N$+C$
```

```
 1750 PRINT "> ";R$(RR)
 1760 GOTO 1440
 1770 IF RR=0 THEN RETURN
 1780 M=0
 1790 M=M+1
 1800 IF M>RR THEN RETURN
 1810 Z$(K)=R$(M)
 1820 IF K=1000 THEN PRINT "OUT OF MEMOR
Y":GOTO 70
 1830 K=K+1
 1840 GOTO 1790
 1850 REM **************************
 1860 REM LIST ALL
 1870 PRINT
 1880 K=0
 1890 K=K+1
 1900 IF Z$(K)="" THEN RETURN
 1910 PRINT Z$(K)
 1920 IF K<1000 THEN 1890
 1930 RETURN
 1940 REM **************************
 1950 REM FORM RULES WITH 'AND' OF THE
          FOLLOWING TYPE:
 1960 REM (X EATS Y IF X IS-BIRD AND
          Y COMES IN BOXES)
 1970 REM X STATEMENT MUST BE IN LIST
PRECEDING Y FOR ALL EXAMPLES TO BE CODED
 1980 REM SPLIT INTO SECTIONS
 1990 J$=MID$(J$,2):REM STRIP "X"
 2000 PRINT TAB(20);"COMPILING RULE"
 2010 J=1
 2020 J=J+1
 2030 IF MID$(J$,J,1)=" " THEN 2060
 2040 IF J<LEN(J$) THEN 2020
```

```
 2050 PRINT "RULE COMPILING ERROR":RETUR
N
 2060 A$=LEFT$(J$,J):REM RELATIONSHIP 1
 2070 J$=MID$(J$,J+7):REM STRIP TO START
 OF SECOND RELATIONSHIP
 2080 J=1:count=0
 2090 J=J+1
 2100 IF MID$(J$,J,1)=" " THEN count=cou
nt+1
 2110 IF count=2 THEN 2140
 2120 IF J<LEN(J$) THEN 2090
 2130 PRINT "RULE COMPILING ERROR":RETUR
N
 2140 B$=LEFT$(J$,J):REM STATEMENT 1
 2150 C$=MID$(J$,J+6):REM STATEMENT 2
 2160 IF C$=" " THEN PRINT "RULE COMPILI
NG ERROR":RETURN
 2170 REM NOW GO THROUGH DATA BASE
 2180 FOR T=1 TO 200
 2190 R$(T)=""
 2200 NEXT T
 2210 R1=0:R2=99
 2220 K=0
 2230 K=K+1
 2240 IF Z$(K)="" THEN 2310
 2250 IF R1=99 OR R2=200 THEN PRINT "MEM
ORY SHORTAGE":GOTO 2310
 2260 LB=LEN(B$)
 2270 IF RIGHT$(Z$(K),LB)=B$ THEN R1=R1+
1:R$(R1)=LEFT$(Z$(K),LEN(Z$(K))-LB)
 2280 LC=LEN(C$)
 2290 IF RIGHT$(Z$(K),LC)=C$ THEN R2=R2+
1:R$(R2)=LEFT$(Z$(K),LEN(Z$(K))-LC)
 2300 IF K<1000 THEN 2230
```

```
 2310 IF R$(100)="" THEN PRINT "STATEMEN
T 2 OF INPUT NOT IN DATA BASE":RETURN
 2320 REM NOW ENCODE RULES
 2330 R1=0:R2=99
 2340 R2=R2+1
 2350 R1=R1+1
 2360 IF R$(R1)>" " AND R$(R2)>" " THEN
Z$(K)=R$(R1)+A$+R$(R2)+" "
 2370 PRINT "> ";Z$(K)
 2380 K=K+1
 2390 IF R$(R2+1)<>"" THEN 2340
 2400 IF R$(R1+1)<>"" THEN 2350
 2410 RETURN
 2420 REM ***************************
 2430 REM ARITHMETIC
 2440 LJ=LEN(J$)
 2450 IF ARITHFLAG<3 THEN GOSUB 2490
 2460 IF ARITHFLAG=3 THEN GOSUB 2890
 2470 IF ARITHFLAG=4 THEN GOSUB 3080
 2480 RETURN
 2490 REM ***** SUM TITLES *****
 2500 J$=MID$(J$,5,LJ-5)
 2510 IF LEFT$(J$,2)="S(" THEN J$=MID$(J
$,3)
 2520 LJ=LEN(J$)
 2530 K=0
 2540 K=K+1
 2550 IF MID$(J$,K,1)=" " THEN A$=LEFT$(
J$,K-1):J$=MID$(J$,K+1):GOTO 2580
 2560 IF K<LJ THEN 2540
 2570 PRINT TAB(12);"ARITHMETIC ERROR":R
ETURN
 2580 LJ=LEN(J$)
 2590 K=0
 2600 K=K+1
```

```
 2610 IF MID$(J$,K,1)=" " THEN B$=LEFT$(
J$,K-1):J$=MID$(J$,K+1):GOTO 2640
 2620 IF K<LJ THEN 2600
 2630 PRINT TAB(12);"ARITHMETIC ERROR":R
ETURN
 2640 LJ=LEN(J$)
 2650 K=0
 2660 K=K+1
 2670 IF MID$(J$,K,1)=")" THEN C$=LEFT$(
J$,K-1):GOTO 2700
 2680 IF K<LJ THEN 2660
 2690 PRINT TAB(12);"ARITHMETIC ERROR (T
OO MANY VARIABLES)":RETURN
 2700 AN=0:BN=0:CN=0
 2710 IF ASC(A$)>58 THEN AN=1
 2720 IF ASC(B$)>58 THEN BN=2
 2730 IF ASC(C$)>58 THEN CN=4
 2740 GUIDE=AN+BN+CN:IF GUIDE=3 OR GUIDE
=5 OR GUIDE=6 THEN 2690
 2750 IF ARITHFLAG=2 THEN 2820:REM TIMES
 2760 IF GUIDE>0 THEN 2790
 2770 IF VAL(A$)+VAL(B$)=VAL(C$) THEN PR
INT "YES":RETURN
 2780 PRINT "NO":RETURN
 2790 IF GUIDE=1 THEN PRINT VAL(C$)-VAL(
B$):GOSUB 40:RETURN
 2800 IF GUIDE=2 THEN PRINT VAL(C$)-VAL(
A$):GOSUB 40:RETURN
 2810 PRINT VAL(A$)+VAL(B$):GOSUB 40:RET
URN
 2820 REM * TIMES *
 2830 IF GUIDE>0 THEN 2860
 2840 IF VAL(A$)*VAL(B$)=VAL(C$) THEN PR
INT "YES":RETURN
 2850 PRINT "NO":RETURN
```

```
 2860 IF GUIDE=1 THEN PRINT VAL(C$)/VAL(
B$):GOSUB 40:RETURN
 2870 IF GUIDE=2 THEN PRINT VAL(C$)/VAL(
A$):GOSUB 40:RETURN
 2880 PRINT VAL(A$)*VAL(B$):GOSUB 40:RET
URN
 2890 REM * LESS *
 2900 NF=0
 2910 IF ASC(J$)<58 THEN NF=1:REM NUMBER
S
 2920 count=0
 2930 K=0
 2940 K=K+1
 2950 IF MID$(J$,K,1)=" " THEN count=cou
nt+1
 2960 IF count=2 THEN 3000
 2970 IF K<LEN(J$) THEN 2940
 2980 PRINT TAB(20);"COMPARISON ERROR"
 2990 RETURN
 3000 B$=MID$(J$,K+1)
 3010 A$=LEFT$(J$,K-6)
 3020 IF NF=1 THEN 3050
 3030 IF A$<B$ THEN PRINT "YES":RETURN
 3040 PRINT "NO":RETURN
 3050 REM * NUMBERS *
 3060 IF VAL(A$)<VAL(B$) THEN PRINT "YES
":RETURN
 3070 PRINT "NO":RETURN
 3080 REM * INT *
 3090 IF RIGHT$(J$,2)="X " THEN 3190
 3100 K=0
 3110 K=K+1
 3120 IF MID$(J$,K,1)=" " THEN 3160
 3130 IF K<LEN(J$) THEN 3110
 3140 PRINT TAB(20);"ARITHMETIC ERROR"
```

```
 3150 RETURN
 3160 A=VAL(LEFT$(J$,K-1))
 3170 IF INT(A)=A THEN PRINT "YES":RETUR
N
 3180 PRINT "NO":RETURN
 3190 K=0
 3200 K=K+1
 3210 IF MID$(J$,K,1)=" " THEN 3240
 3220 IF K<LEN(J$) THEN 3200
 3230 PRINT TAB(20);"ARITHMETIC ERROR":R
ETURN
 3240 PRINT INT(VAL(LEFT$(J$,K-1)))
 3250 RETURN
 3260 REM **************************
 3270 DEF PROCinitialise
 3280 CLS
 3290 DIM Z$(1000),R$(200)
 3300 ENDPROC
```

SSLISP

```
   10 REM S.S.LISP
   20 MODE 6:VDU 19,0,4;0;:GOTO 3450:REM
 INITIALISE
   30 REM ******************************
   40 A$=MID$(A$,E):A$=LEFT$(A$,LEN(A$)-
1)
   50 RETURN
   60 REM ******************************
   70 PRINT
   80 NN=0
   90 INPUT ": "A$
  100 IF A$="" THEN END:REM JUST PRESS
                        <RETURN> TO END RUN
```

```
110 REM ******************************
120 FOR J=1 TO 12
130 X(J)=0:Y(J)=0:Z(J)=0
140 NEXT J
150 REM ******************************
160 R=0:S=0:T=0:CFIRST=0:CSECND=0:EDGE
=0
170 FOR J=1 TO LEN(A$)
180 B$=MID$(A$,J,1)
190 IF B$="(" THEN S=S+1:Z(S)=J:IF T=0
 THEN CFIRST=J
200 IF B$=")" THEN T=T+1:Y(T)=J:IF CSE
CND<>0 AND EDGE=0 THEN EDGE=J-1
210 IF T=1 AND B$=")" THEN CSECND=J
220 IF B$=" " THEN R=R+1:X(R)=J
230 NEXT J
240 IF S=T THEN 300:REM ( ) BALANCE
250 IF S<T THEN PRINT" -> MISSING ("
260 IF S>T THEN PRINT" -> MISSING )"
270 INPUT"  +   "B$
280 A$=A$+B$
290 GOTO 120
300 IF NWDS=0 OR NN=1 THEN 370
310 M$=LEFT$(A$,X(1)-1)
320 FOR J=1 TO NWDS
330 IF M$=N$(J) THEN A$=O$(J)+MID$(A$,
LEN(N$(J))+1)
340 NEXT J
350 NN=1
360 GOTO 120
370 FLAG=0:B$="NIL"
380 IF LEFT$(A$,5)="CAR (" THEN FLAG=1
390 IF LEFT$(A$,5)="CDR (" THEN FLAG=2
400 IF LEFT$(A$,6)="CONS (" THEN FLAG=
3
```

```
410 IF LEFT$(A$,6)="ATOM (" THEN FLAG=
4
420 IF LEFT$(A$,4)="EQ (" THEN FLAG=5
430 IF LEFT$(A$,6)="NULL (" THEN FLAG=
6
440 IF LEFT$(A$,8)="MEMBER (" THEN FLA
G=7
450 IF LEFT$(A$,6)="MEMQ (" THEN FLAG=
8
460 IF LEFT$(A$,8)="APPEND (" THEN FLA
G=9
470 IF LEFT$(A$,9)="REVERSE (" THEN FL
AG=10
480 IF LEFT$(A$,7)="EQUAL (" THEN FLAG
=11
490 IF LEFT$(A$,6)="LIST (" THEN FLAG=
12
500 IF LEFT$(A$,8)="DEFINE (" THEN FLA
G=13
510 IF LEFT$(A$,6)="ADD1 (" THEN FLAG=
14
520 IF LEFT$(A$,6)="SUB1 (" THEN FLAG=
15
530 IF LEFT$(A$,7)="ZEROP (" THEN FLAG
=16
540 IF LEFT$(A$,12)="DIFFERENCE (" THE
N FLAG=17
550 IF LEFT$(A$,6)="EXPT (" THEN FLAG=
18
560 IF LEFT$(A$,5)="MAX (" THEN FLAG=1
9
570 IF LEFT$(A$,5)="MIN (" THEN FLAG=2
0
580 IF LEFT$(A$,6)="PLUS (" THEN FLAG=
21
```

```
  590 IF LEFT$(A$,7)="MINUS (" THEN FLAG
=22
  600 IF LEFT$(A$,10)="QUOTIENT (" THEN
FLAG=23
  610 IF LEFT$(A$,7)="RECIP (" THEN FLAG
=24
  620 IF LEFT$(A$,11)="REMAINDER (" THEN
 FLAG=25
  630 IF LEFT$(A$,7)="TIMES (" THEN FLAG
=26
  640 IF LEFT$(A$,10)="GREATERP (" THEN
FLAG=27
  650 IF LEFT$(A$,7)="LESSP (" THEN FLAG
=28
  660 IF LEFT$(A$,8)="MINUSP (" THEN FLA
G=29
  670 IF LEFT$(A$,9)="NUMBERP (" THEN FL
AG=30
  680 IF LEFT$(A$,6)="ONEP (" THEN FLAG=
31
  690 IF FLAG>13 THEN 720
  695 IF FLAG=0 THEN 70:REM KEYWORD NOT
                           RECOGNISED
  700 ON FLAG GOSUB 840,890,970,1110,120
0,1330,1380,1510,1700,1760,2000,2130,330
0
  710 GOTO 760
  720 IF FLAG>24 THEN 750
  730 ON (FLAG-13) GOSUB 2180,2230,2280,
2350,2560,2600,2790,2820,2870,2920,2960,
740
  740 GOTO 760
  750 ON (FLAG-24) GOSUB 3030,3070,3120,
3160,3200,3250,2280
  760 IF FLAG<>0 THEN GOSUB 780
  770 GOTO 70
```

```
  78Ø REM ** RETURN ANSWER **
  79Ø PRINT "    VALUE IS ..."
  8ØØ IF B$<>"()" THEN PRINT"     ";B$
  81Ø IF B$="()" THEN PRINT"    NIL"
  82Ø RETURN
  83Ø REM ******************************
**
  84Ø REM                 ** CAR **
  85Ø IF S=2 THEN B$=MID$(A$,Z(2)+1,X(2)
-Z(2))
  86Ø IF S>2 THEN B$=MID$(A$,CFIRST,CSEC
ND-CFIRST+1)
  87Ø RETURN
  88Ø REM ******************************
**
  89Ø REM                 ** CDR **
  9ØØ GOSUB 84Ø
  91Ø LB=LEN(B$)+7
  92Ø B$="("+MID$(A$,LB,EDGE-1)
  93Ø IF RIGHT$(B$,2)="))" THEN B$=LEFT$
(B$,LEN(B$)-1)
  94Ø IF MID$(B$,2,1)=" " THEN B$="("+MI
D$(B$,3)
  95Ø RETURN
  96Ø REM ******************************
**
  97Ø REM                 ** CONS **
  98Ø B$=MID$(A$,7,LEN(A$)-1)
  99Ø J=Ø
 1ØØØ IF LEFT$(B$,1)="(" THEN J=1
 1Ø1Ø J=J+1
 1Ø2Ø IF MID$(B$,J,1)="(" THEN 1Ø5Ø
 1Ø3Ø IF J<LEN(B$) THEN 1Ø1Ø
 1Ø4Ø B$="    > CONS ERROR <":RETURN
 1Ø5Ø LB=LEN(B$)-1
```

```
 1060 B$="("+LEFT$(B$,J-1)+MID$(B$,J+1)
 1070 B$=LEFT$(B$,LB)
 1080 IF RIGHT$(B$,2)=" )" THEN B$=LEFT$
(B$,LEN(B$)-2)+" )"
 1090 RETURN
 1100 REM ******************************
**
 1110 REM                 ** ATOM **
 1120 A$=MID$(A$,7,LEN(A$)-1)
 1130 J=0:B$="NIL"
 1140 J=J+1
 1150 IF MID$(A$,J,1)=" " OR MID$(A$,J,1
)="(" THEN RETURN
 1160 IF J<LEN(A$) THEN 1140
 1170 B$="T"
 1180 RETURN
 1190 REM ******************************
**
 1200 REM                   ** EQ **
 1210 E=5:GOSUB 40
 1220 J=0
 1230 J=J+1
 1240 IF MID$(A$,J,1)=")" THEN RETURN
 1250 IF MID$(A$,J,1)=" " THEN 1280
 1260 IF J<LEN(A$) THEN 1230
 1270 RETURN
 1280 C$=LEFT$(A$,J-1)
 1290 A$=MID$(A$,J+1)
 1300 IF C$=A$ THEN B$="T"
 1310 RETURN
 1320 REM ******************************
**
 1330 REM                 ** NULL **
 1340 IF A$="NULL ()" THEN B$=" ILLEGAL
- NULL NEEDS ARGUMENT"
```

```
 1350 IF A$="NULL (())" THEN B$="T"
 1360 RETURN
 1370 REM *******************************
*
 1380 REM             ** MEMBER **
 1390 C$=MID$(A$,9)
 1400 J=1
 1410 J=J+1
 1420 IF MID$(C$,J,1)=")" OR MID$(C$,J,1
)="(" THEN D$=LEFT$(C$,J):GOTO 1450
 1430 IF J<LEN(C$) THEN 1410
 1440 RETURN
 1450 J=LEN(D$)
 1460 J=J+1
 1470 IF MID$(C$,J,LEN(D$))=D$ THEN C$=L
EFT$(C$,LEN(C$)-1):GOTO 1630
 1480 IF J<LEN(C$) THEN 1460
 1490 RETURN
 1500 REM ******************************
 1510 REM             ** MEMQ **
 1520 C$=MID$(A$,7)
 1530 J=0
 1540 J=J+1
 1550 IF MID$(C$,J,1)=" " THEN 1580
 1560 IF J<LEN(A$) THEN 1540
 1570 RETURN
 1580 D$=LEFT$(C$,J)
 1590 C$=MID$(C$,J+2)
 1600 C$=LEFT$(C$,LEN(C$)-2)+" "
 1610 J=0
 1620 J=J+1
 1630 IF MID$(C$,J,LEN(D$))=D$ THEN B$="
 ("+MID$(C$,J):GOTO 1660
 1640 IF J<LEN(C$) THEN 1620
 1650 RETURN
 1660 B$=LEFT$(B$,LEN(B$)-1)+")"
```

```
1670 IF RIGHT$(B$,3)=")))" THEN B$=LEFT$(B$,LEN(B$)-1):GOTO 1670
1680 RETURN
1690 REM *****************************
1700 REM          ** APPEND **
1710 B$=MID$(A$,9)
1720 B$=LEFT$(B$,Y(1)-9)+" "+MID$(B$,Z(3)-7)
1730 B$=LEFT$(B$,LEN(B$)-1)
1740 RETURN
1750 REM *****************************
1760 REM         ** REVERSE **
1770 B$=""
1780 A$=MID$(A$,11):A$=LEFT$(A$,LEN(A$)
1790 CT=0
1800 J=0
1810 J=J+1:IF J>LEN(A$) THEN 1920
1820 IF MID$(A$,J,1)=" " THEN 1850
1830 IF MID$(A$,J,1)="(" THEN 1860
1840 GOTO 1810
1850 CT=CT+1:G$(CT)=LEFT$(A$,J-1):A$=MID$(A$,J+1):GOTO 1800
1860 J=J+1:IF MID$(A$,J,2)="))" THEN 1980
1870 IF MID$(A$,J,1)=")" THEN 1910
1880 IF J=LEN(A$) THEN 1900
1890 GOTO 1860
1900 CT=CT+1:G$(CT)=A$+")":GOTO 1930
1910 CT=CT+1:G$(CT)=LEFT$(A$,J):A$=MID$(A$,J+1):GOTO 1800
1920 CT=CT+1:G$(CT)=A$
1930 FOR M=CT TO 1 STEP -1
1940 B$=B$+G$(M):IF M>1 THEN B$=B$+" "
1950 NEXT M
1960 B$="("+B$+")"
```

```
 1970 RETURN
 1980 CT=CT+1:G$(CT)=LEFT$(A$,J+1):A$=MI
D$(A$,J+2):GOTO 1800
 1990 REM ****************************
 2000 REM          ** EQUAL **
 2010 E=8:GOSUB 40
 2020 M=ASC(A$):IF M>47 AND M<58 THEN 23
70
 2030 J=0
 2040 J=J+1
 2050 IF MID$(A$,J,2)=") " THEN J=J+1:GO
TO 1280
 2060 IF MID$(A$,J,3)=")))" THEN 2100
 2070 IF MID$(A$,J,2)="))" THEN 2110
 2080 IF J<LEN(A$) THEN 2040
 2090 RETURN
 2100 C$=LEFT$(A$,J+2):A$=MID$(A$,J+4):G
OTO 1300
 2110 C$=LEFT$(A$,J+1):A$=MID$(A$,J+3):G
OTO 1300
 2120 REM ****************************
 2130 REM             ** LIST **
 2140 E=7:GOSUB 40
 2150 B$="("+A$+")"
 2160 RETURN
 2170 REM ****************************
 2180 REM             ** ADD1 **
 2190 E=7:GOSUB 40
 2200 B$=STR$(VAL(A$)+1)
 2210 RETURN
 2220 REM ****************************
 2230 REM             ** SUB1 **
 2240 E=7:GOSUB 40
 2250 B$=STR$(VAL(A$)-1)
 2260 RETURN
```

```
2270 REM ****************************
2280 REM          ** ZEROP **
2290 IF FLAG=16 THEN E=8
2300 IF FLAG=31 THEN E=7
2310 GOSUB 40
2320 IF (A$="0" AND FLAG=16) OR (A$="1"
AND FLAG=31) THEN B$="T"
2330 RETURN
2340 REM ****************************
2350 REM       ** TWO ARGUMENTS **
2360 E=13:GOSUB 40
2370 J=0
2380 J=J+1
2390 IF MID$(A$,J,1)=" " THEN 2420
2400 IF J<LEN(A$) THEN 2380
2410 B$=" *ERROR - ONLY ONE ARGUMENT*":
RETURN
2420 P=VAL(LEFT$(A$,J-1))
2430 Q=VAL(MID$(A$,J+1))
2440 IF FLAG=17 THEN B$=STR$(P-Q):RETUR
N
2450 IF FLAG=23 OR FLAG=25 THEN B=P/Q
2460 IF FLAG=25 THEN B=INT(.5+Q*(B-INT(
B))*1000)/1000
2470 IF FLAG=18 THEN B=P^Q
2480 IF FLAG=11 AND P=Q THEN B$="T"
2490 IF FLAG=27 AND P>Q THEN B$="T"
2500 IF FLAG=28 AND P<Q THEN B$="T"
2510 IF FLAG=32 THEN B=P-Q
2520 IF FLAG=11 OR FLAG>26 THEN RETURN
2530 B$=STR$(B)
2540 RETURN
2550 REM ****************************
2560 REM          ** EXPT **
2570 E=7:GOSUB 40
2580 GOTO 2370
```

```
2590 REM ******************************
2600 REM    ** MAX (MIN PLUS TIMES) **
2610 F$=LEFT$(A$,3):A$=MID$(A$,6)
2620 CT=0:FLAG=0
2630 IF F$="TIMES" THEN CT=1
2640 J=0
2650 J=J+1
2660 IF MID$(A$,J,1)=" " THEN 2690
2670 IF J<LEN(A$) THEN 2650
2680 IF J=LEN(A$) THEN FLAG=1
2690 P=VAL(LEFT$(A$,J-1)):IF FLAG=0 THE
N A$=MID$(A$,J+1)
2700 IF F$<>"PLUS" AND CT=0 THEN CT=P
2710 IF F$="MAX" AND P>CT THEN CT=P
2720 IF F$="MIN" AND P<CT THEN CT=P
2730 IF F$="PLUS" THEN CT=CT+P
2740 IF F$="TIMES" THEN CT=CT*P
2750 IF FLAG=0 THEN 2640
2760 B$=STR$(CT)
2770 RETURN
2780 REM ******************************
2790 REM             ** MIN **
2800 GOTO 2610
2810 REM ******************************
2820 REM            ** PLUS **
2830 F$="PLUS"
2840 A$=MID$(A$,7)
2850 GOTO 2620
2860 REM ******************************
2870 REM            ** MINUS **
2880 E=8:GOSUB 40
2890 B$=STR$(-VAL(A$))
2900 RETURN
2910 REM ******************************
2920 REM       ** QUOTIENT **
```

```
2930 E=11:GOSUB 40
2940 GOTO 2370
2950 REM ******************************
2960 REM        ** RECIP **
2970 E=8:GOSUB 40
2980 IF A$="0" THEN B$="DIVISION BY ZER
O ILLEGAL":RETURN
2990 B=1/(VAL(A$))
3000 B$=STR$(B)
3010 RETURN
3020 REM ******************************
3030 REM        ** REMAINDER **
3040 E=12:GOSUB 40
3050 GOTO 2370
3060 REM ******************************
3070 REM          ** TIMES **
3080 F$="TIMES"
3090 A$=MID$(A$,8)
3100 GOTO 2620
3110 REM ******************************
3120 REM        ** GREATERP **
3130 E=11:GOSUB 40
3140 GOTO 3270
3150 REM ******************************
3160 REM          ** LESSP **
3170 E=8:GOSUB 40
3180 GOTO 2370
3190 REM ******************************
3200 REM          ** MINUSP **
3210 E=9:GOSUB 40
3220 IF VAL(A$)<0 THEN B$="T"
3230 RETURN
3240 REM ******************************
3250 REM         ** NUMBERP **
3260 A$=MID$(A$,10)
```

```
 3270 IF ASC(A$)>44 AND ASC(A$)<58 THEN
B$="T"
 3280 RETURN
 3290 REM ******************************
 3300 REM          ** DEFINE **
 3310 A$=MID$(A$,9)
 3320 F$=LEFT$(A$,X(2)-9)
 3330 G$=MID$(A$,X(4)-6)
 3340 J=0
 3350 J=J+1
 3360 IF MID$(G$,J,1)=" " THEN 3390
 3370 IF J<LEN(G$) THEN 3350
 3380 B$=" DEFINE ERROR":RETURN
 3390 G$=LEFT$(G$,J-1)
 3400 NWDS=NWDS+1
 3410 O$(NWDS)=G$:N$(NWDS)=F$
 3420 B$=F$
 3430 RETURN
 3440 REM ******************************
 3450 REM        ** INITIALISE **
 3460 CLS
 3470 DIM G$(20),O$(20),N$(20),X(12),Y(1
2),Z(12)
 3480 NWDS=0:REM COUNT OF USER DEFINED
                 'NEW WORDS'
 3490 GOTO 70
```

Appendices:

A — Bayes' Theorem

B — Databases

C — Fuzzy Logic Rules

D — Weather Data

E — References

F — Further Reading

Appendix A Bayes' Theorem and Probabilities

The Reverend Thomas Bayes, who lived from 1702 to 1871, was a Presbyterian minister and a very skilled mathematician. Expert systems had barely been dreamed of in his time (and their skills would probably have been ascribed to the devil). Despite this, his work finds ready application in any field (such as expert systems) where the probabilities of events taking place need to be modified as additional information is gathered.

One suspects Bayes would not be pleased. He started on the road which lead to the development of his theorem when he voiced the admirable hypothesus that the existence of the Almighty could be proved by examination of the mathematical beauties in the world which He created. I will, he said, prove "that the Principle End of the Divine Providence . . . is the Happiness of His Creatures" and I will do it through mathematics.

Unfortunately, the further he got into his studies, the more alarmed Bayes became by the implications of his discoveries. He finally closed the book on his work, and decided that it could not be published in his lifetime.

However, the work he did was solid, and lies at the heart of modern decision-making theory. It is often called Bayesian decision theory in his honor. His theorem gives us a mathematically sound way of evaluating new information, and of using it to modify earlier estimates — based on limited data — of the probability that particular outcomes will occur. It allows us to act on partial knowledge — as an expert system will often have to do — and then evaluate and revise our decisions as more data comes in.

To understand Bayes' Theorem, and see how it could be of value to you when developing your own expert systems, we need to know a little about probability.

The chance of an event occurring is the number of successful outcomes divided by the number of possible outcomes. That is, the chance of a coin landing showing the head is 1 (the number of outcomes where the event required occurs) divided by 2 (the number of possible outcomes).

To put that into an equation, where P(outcome) is used to indicate the probability of that outcome, we could write:

```
                number of successful outcomes
P(outcomes)=--------------------------------
                number of possible outcomes
```

For our coin-tossing example, we could write:

```
              1          1
P(head)=---------  =  -----  = 0.5
            1 + 1        2
```

If more than one event is being examined (as opposed to the coin tossing situation where we only interested in the single toss of a single coin), the events are either *mutually exclusive* (if it is raining, it cannot be not raining) or *nonmutually exclusive* (it can be cold *and* foggy).

Mutually Exclusive Events

The probability for mutually exclusive events is the probability that outcome x *or* outcome y will occur. The probabilities in this case are *added* together, as follows:

```
P(outcomeX OR outcomeY)
              = P(outcomeX) + P(outcomeY)
```

Imagine we are throwing a single die. We first want to know what is the chance of it landing showing a three:

```
                    1
      P(three) = -----
                    6
```

The chance of it landing showing a five, P(five), is the same, one in six. Now the die cannot land showing *both* a three and a five, so the events are mutually exclusive. This means that the chance of the die landing showing *either* a three or a five can be expressed as:

```
   P(three OR five)=P(three) + P(five)
                  = 1/6 + 1/6 = 2/6 = 1/3
```

In other words, there is one chance in three that we will get a three OR a five on a single throw of the die.

Now, when we throw the die, it either comes up three or five, or it doesn't. What are the chances of it falling NOT showing a three or a five. Fairly obviously, the chance is 2/3 or 1 — 1/3. When we add the probability that an event *will* occur to the probability that it *will not*, it must add up to 1:

```
    P(event) + P(not event) = 1
```

. . . which is the same as . . .

```
P(not event)=1 - P(event)
```

Events which are not Mutually Exclusive

Imagine that you are holding a key which fits just one of six boxes which are on the table in front of you. You do not know which box it fits. However, you do know what is in the boxes: a black block, a black ball, a green block, a red ball, a black toothbrush, and a blue inflatable mouse. You try the key on each box, until it opens one. What is the chance that the box you open contains something black or a ball? Obviously, these events are not mutually exclusive, as there is one object (the black ball) which satisfies both criteria.

However, opening a box which contains a ball does not necessarily mean this ball will be black. But because the possibility does exist of opening a box which satisfies both conditions, we need to reduce the probability that one condition will be satisfied by the chance that both will be satisfied.

The chance of getting a black object is 3/6, and the chance of getting a ball is 2/6. The equation for selecting either a ball or a black object (that is, the probability of one or more of two events that are not mutually exclusive) is:

```
P(black OR ball)=
  P(black) + P(ball) - P(black AND ball)
```

In words, this equation means: The probability of choosing a black object or a ball is equal to the probability of choosing a black object plus the probability of choosing a ball, minus the probability of choosing a black object which is also a ball.

```
P(black OR ball) = 3/6 + 2/6 - 1/6
                 = 5/6 - 1/6
                 = 4/6
                 = 2/3
```

Statistical Independence

You may wonder how all this relates to Reverend Bayes. It does, and it will all be clear in due course. Bayes' work cannot be explained until we've been through our probabilities.

If, when you hit someone in the face with your mighty, hammer-like fist, they later develop a black eye, we say the events are statistically dependent. The probability of the second event occurring (the eye going black) is closely related to the probability of you smacking someone hard enough in the eye with your fist. However, just because two events occur in a row does not mean they are always statiscally dependent. Throw a coin. It lands heads. Throw it again. The probability of getting a head on the second throw is totally independent of the result of the first throw.

If we call the probability of getting a four when we throw a die P(four), and the chance of a six P(six), the chance of getting a four on the first throw, and a six on the second, can be expressed as follows (where P(four & six) means the probability of four and six occurring together or in succession):

```
P(four & six) = P(four) x P(six)
```

The chance of getting a four is 1/6; and the chance of getting a six is also 1/6, so the chance of getting a four followed by a six is 1/6 times 1/6, that is, 1/36. The chance of getting a four, followed by a six, followed by a three, is 1/6 times 1/6 times 1/6, or 1/216. As an equation, we'd have:

```
P(four & six & three) =
        P(four) x P(six) x P(three)
```

The chance that this will *not* occur (that is, we will not throw a four, followed by a six, followed by a three) is one minus the chance that the event will occur:

```
P(NOT (four & six & three))=
          1 - (P(four) x P(six) x P(three))
```

This way of working out the chance of an event *not* occurring is true for any situation; simply subtract the chance of an event occurring from one. If the chance of getting a six when throwing a die, P(six), is 1/6, the chance of not throwing a six is 1 — 1/6, that is, 5/6.

Conditional Probability

Conditional probability refers to the chance of event Y occurring, given that event X *has* occurred. It is written as P(Y/X). If the events are *statistically independent*, the probability of event Y, given that event X has occurred, is (perhaps surprisingly) simply the probability of event Y occurring, P(Y).

Why should this be? If we throw a die, the chance of it coming up with a six is 1/6. If we throw it again, the probability for that throw coming up a six again is still 1/6. One throw of a die does not influence what happens on following throws.

Note that in this case, that of statistical independence, we are asking what the chance of event Y occuring, given that event X *has* occurred; we are not asking what is the chance of event Y AND event X occurring. We are asking, given that event X has occurred (we have thrown a six), what the probability is of event Y (throwing a six).

Statistical Dependence

Things get a little more involved, I'm afraid to say, when the chance of a second event is related to the probability of a first event.

Conditional Probability

Let's imagine a situation in which we are holding a key which fits one of nine boxes sitting in front of us. Five of the boxes contain books written by Tim Hartnell, two contain books by Dr Rodnay Zaks, and the other two hold books from Grace Murray Hopper (Commodore Hopper, at the time of writing, was the US Navy's most senior woman officer, and its oldest serving officer, despite trying to retire several times; Hopper designed the computer language COBOL).

The chance of the box which is opened by the key containing any one of the books is 1/9. There are nine books and any one of them has the same probability as being in the box which the key fits as any other. However, suppose the box which can be opened contains a book by a male author. The probability of this occurring is 7/9. What is the probability that it will be one written by Rodnay Zaks? We can write this as P(Z:M), the probability that the book will be by Zaks, P(Z), given that the book is by a male author, P(M).

We know the book is by a male author. To work out the chance that this book is by Zaks, we ignore the books written by Hopper, as these cannot be involved in this situation. We know there are seven books by males, two of which are by Zaks. To find the probability of Hartnell and Zaks within this seven, we divide the number of books by each author, by the total number of books by male authors:

```
P(Z¦M) = 2/7
P(H¦M) = 5/7
```

These probabilities add up, as they should, to one. The probability of a book being by Zaks, given that the book is by a male author, is 2/7; and the probability of a book being by Hartnell, given that the book is by a male author, is 5/7.

There is a higher probability, given that the book is by a male author, of it being by Hartnell than by Zaks. To work out the probability of the book being by Zaks, given that the book is by a male author, P(Z/M), we divide the probability of Zaks by the probability of male, P(M), where P(M) equals P(Z) plus P(H):

```
         P(Z)   P(2/9)   .2222
P(Z|M) = ---- = ------ = ----- = .286
         P(M)   P(7/9)   .7777
```

Now, as a check on this, we can reason that there are seven books by male authors. If the book we have is by a male, there are two chances (out of the seven) that it is by Zaks. Therefore, if our method of working out P(Z/M) above is correct, it should give the same answer as 2/7 (which is the chance that out of a set of seven books, two of which are by Zaks, a Zaks book is chosen). In fact, we find that this is the case, as it should be.

Conditional probability, then, when outcomes are statistically dependent, can be expressed as:

```
P(Y|X) = P(YX)/P(X)
```

Back to Bayes

This, at long last, brings us to the position where we can appreciate the work of the good Reverend. You may recall, at the start of this section, I said that Bayes' Theorem gives us a way to use information obtained later to modify earlier estimates, based on limited data, of the probability that particular outcomes will occur.

Imagine we have two boxes, each of which contain 25 wooden balls. In one box (B1), there are 14 black balls, and 11 red ones. In the second box (B2), there are 19 black balls and six red ones. You choose a box at random, plunge in your hand, and bring out a ball. It is black. What is the chance that you drew the ball out from B2?

The chance of getting either B1 or B2 is 1/2 (0.5). The chance of getting a black from B1 is 14/25 (0.56), and the chance of getting a black from B2 is 19/25 (0.76). The chance of getting a B1 AND a black ball is 0.5 times 0.56 (0.28) and the probability of B2 AND a black ball is 0.5 times 0.76 (0.38). The chance of getting a black ball at all is the sum of these two probabilities, 0.28 plus 0.38 (0.66). The chance, then, that we got the ball from B1 is P(B1, black)/P(black), or 0.28 divided by 0.66, or approximately 0.424 and the chance we got the ball from B2 is 1 — 0.424 or 0.576 (as we could only have got it from B1 or B2, the sum of the probabilities must equal 1).

What does this tell us? What significance does the 0.424 or 0.576 have? Before we reached into a box and drew out a ball, we would have said the chance of the ball coming from B1 or B2 would be 0.5, but now — after choosing only one ball — we can say that there is a higher chance that the ball came from B2 rather than from B1.

Put the ball back in B2. Shuffle the boxes(!). Choose a box at random, and choose a ball from it. Imagine that we've managed to get another black ball. Can we say, with any confidence, what the chances are that, once again, we've taken it from B2? The probability that the two blacks came from B1 is 0.5 times 0.56 times 0.56 (0.159) and the probability that the two blacks came from B2 is 0.5 times 0.76 times 0.76 (0.289). If we add these together, we get the chance of getting two blacks in a row (0.159 plus 0.289 is 0.448).

Now, how do we work out what the probability is that we took the second black ball from B2?

The chance of getting two blacks in a row from B1 is the chance of choosing B1 and getting two blacks from it (0.159) divided by the chance of getting two blacks in a row (0.448), which is 0.355. The probability of getting two blacks in a row from B2 should be 1 — 0.355, or 0.645. Let's see if it is. The chance of choosing B2 and getting two blacks from it (0.289) divided by the chance of getting two blacks in a row (0.448), which is 0.645, just as we predicted.

Now where are we? Do we really know anything more than we did at the beginning? We started the whole black ball/box process with only the information that there was one chance in two (0.5) that we would choose B1 or B2. After selecting a single ball, which turned out to be black, we were able to say that the the probability that it came from B1 was 0.424 and the chance that it came from B2 was 0.576. We replaced the ball, shuffled the boxes, and drew another one. It was black again. We did our sums again, and decided that the chance that the ball came from B1 was 0.355 and it was 0.645 that it came from B2. This allows us to say that if we chose two balls in a row (replacing the first ball before choosing a second one), and they were both black, the probability that they came from B2 is 0.645.

Appendix B
Databases

The power of expert systems in the future will be due to the efficiency and intelligence of the inference engine, and the quality and breadth of the knowledge base the system can access.

The information the knowledge base can hold will be, in the short term, essentially textual in nature, as our current computers are far better at manipulating the symbols which represent text than they are at playing with more complex symbols such as those which encode an animated color picture. At present, about 55% of material handled by a typical business is text, with 30% raw data and the balance of around 15% being made up of image information.

When setting up a database for an expert system to access, there are three things which may need consideration. These are:

— the cost of holding the information

— the speed of communicating the information

— the quality of the information held

The sheer *size* of the information to be manipulated must be also be taken into account.

The Cost of Data Storage

The cost of holding information has shrunk dramatically over the past thirty years, as this chart — which shows the rough cost of holding a foolscap sheet of information — convincingly illustrates:

YEAR	IN MAIN MEMORY	ON-LINE DIRECT ACCESS
1950	$225,000,000.00	—
1960	$70,000.00	$12,000.00
1970	$11,500.00	$2,500.00
1975	$2,750.00	$250.00
1980	$275.00	$20.00
1983	$120.00	$15.00
1985 (estimated)	$12.00	0.75
1990 (estimated)	$2.00	0.12

These last two figures are likely to prove to be very conservative estimates. By 1987, electronic storage will be the cheapest method available for holding *text*, and will, of course, offer the additional benefits of immediate accessibility, fast searching and rapid communication. Two years after this, eletronic storage offering full color, random access and animation, will be the cheapest way of holding *image* information.

The Characteristics of Data within Databases

What are the characteristics of large quantities of data, such as those which will be held in knowledge bases? There are three levels. The first is that of *pure data*, a set of unconnected facts. The second level is that of *structured*, or categorised knowledge, held in a form in which it can be handled. This structure is vital if any sense is to be made of the information. The third — and presumably, in many cases, the most valuable level — is that of '*associated* knowledge', in which not only the information is held. In addition, the knowledge base contains knowledge about the *relationships* of items within the data base.

The most important decisions we'll be making over the next decade relate to the *way* in which knowledge, and its interelationships, will be held. There are many ways in which knowledge bases, and information about relationships of items within those bases, can be stored. And the kind of decisions we make now on such bases will lock us into using those forms of organisation perhaps forever. A related current problem is the sheer magnitude of the task of converting 'old databases' (held in such forms as paper files) into electronic forms.

In Britain for example, the Department of Health and Social Security ('welfare') currently has the two problems to cope with. It has some 26.5 million names in active files, related by such things as geographic location, and a massive 'rule base' containing such information as how payments are to be made, to whom, and under what circumstances. According to the DHSS, current technology is unable to handle the rule base. However, the current methods are no longer adequate, so even before the necessary technology exists, decisions need to be made on how information will be handled so that when the technology is available it will be not handicapped unduly by being locked into primitve forms. This is indicative of the kind of problem facing much of the current organisation of knowledge.

How Information is Handled

There are three main components of the way in which information is currently handled. The major component, at present, is the database, followed by the work now going on in expert systems. The third component, which I'll discuss in a moment, is less well-defined, but still forms an important part of the current means in which knowledge is held.

Databases began as simple, structured two-dimensional arrays, and from these have evolved into what are now called 'relational databases'. The major problem with a database is that you need to know *in advance* how information will be organised. This, as a moment's thought will show, can severely limit the effectiveness and flexibility of a database.

The second component in the way knowledge is currently handled is the building up of expert bases for handling by inference engines.

The third component can, perhaps, be best enclosed by the term 'transfer technology'. This is the ability to take skills from one area, and apply them in other areas, or in conjunction with other technologies (such as teaching, computer-based training and the use of 'interactive video', where laser disks may well soon be the most important component).

The laser disk neatly encapsulates one of the current problems and potential of the massive amounts of information we can now store. At present, a laser disk can hold around 55,000 still images; this is about 700 Megabyte of data. They cost less than $5.00 each to mass-produce. These are quite amazing figures. Disks are under development which can be written to by the user, while at present they are read only. The statistics show the potential. The problem lies in the *management* of all that data. In contrast to the majority of textual data, visual imagery is not generally clearly categorisable. However, the vast cross-referencing potential of the computer, coupled with its unique ability to handle data of the density held within a laser disk, shows that video disks could only have evolved, and only make sense, when used with a computer to control and organise the information.

The *quality* of the information held within computer data banks today should also be examined. Most of the information is currently managed by mathematical logic (using operators such as $+$, $-$, $*$, $<$ and $>$). Words are treated as numbers. This means they have — to the computer — a number value, but no meaning. That is, although the information itself can be manipulated as numbers, the system itself cannot have the faintest idea of what it is working with.

If we could produce a system which uses words as words, manipulated by logic, and we might well be several steps closer to an 'aware' machine.

Already it is happening, to some extent. Greater processing power, and lower cost storage, means it is now possible to develop systems based on verbal (semantic) logic. Such a system recognises non-mathematical relationships between words, such as the association which exists between such pairs as father/son, big/large and New York/City, as you've seen in the material in this book on HASTE, EASLE, PROLOG-A and SSLISP.

Appendix C
Fuzzy Logic Rules

Fuzzy logic uses the operators AND, OR and NOT:

NOT: **Given two opposite conditions, the probability of one state is (1 — probability) of the opposite state**

AND: **This takes the lower of two (or more) figures, so if one is 0.3 and the other is 0.5, ANDing them gives 0.3**

OR: **This takes the higher of two (or more) figures, so if one is 0.3 and the other is 0.5, ORing them gives 0.5**

Note that this way of determining values in an AND or OR situation is largely traditional. Some people argue that an AND should be the *multiple* of probability-one and probability-two. Using the traditional method seems to work in practice, and given the largely empirical way in which the quality of output of such a system must be assessed in many cases, the fact that it works is really all that matters.

NOT p1 ------> 1 −p1

p1 AND p2 ------> MIN(p1,p2)

p1 OR p2 ------> MAX(p1,p2)

Appendix D
Weather Data

This is the raw data used for the weather prediction section of FUZZY RITA. As you can see, not all the figures were used. You might like to try the system yourself, making use of information which my sample run ignored.

DATE	M.S.L. BAROM 9 AM MBS	TEMPERATURE		RH AT 3PM (%)	WIND AT 3PM KM/H		MAX GUST KM/H		SUN. HOURS LAV.	EVAP. 24 HRS. END. 9AM (mm)	RAINFALL 24 HRS. END. 9AM (mm)
		MIN (°C)	MAX (°C)								
1	1011.6	11.0	25.5	31	SE	07	SW	28	13.5	3.4	0
2	1006.7	12.6	27.6	29	SW	11	NW	46	10.0	4.8	0
3	1012.4	10.8	16.5	52	S	13	SW	57	12.7	5.6	2.6
4	1014.7	9.1	15.9	74	W	15	W	72	0.0	4.0	1.6
5	1016.5	11.0	16.8	56	SSE	11	WSW	45	0.7	1.2	7.2
6	1018.1	13.0	17.9	60	S	07	SSW	26	1.2	4.2	0
7	1017.9	12.8	23.1	51	S	13	S	43	11.0	2.4	0
8	1016.7	9.9	22.8	38	SE	19	ESE	50	13.6	3.8	0
9	1013.9	11.6	24.2	38	SE	09	SE	45	13.7	6.0	0
10	1010.8	10.5	27.7	22	E	06	SE	45	11.8	5.6	0
11	999.0	15.9	21.3	60	Calm		NW	50	4.5	6.2	2.6
12	1002.6	12.2	23.5	30	W	26	WNW	63	9.8	3.4	2.6
13	1005.9	13.0	17.9	86	NNW	06	W	30	2.6	4.0	0
14	1000.0	12.5	24.2	84	ENE	04	W	61	3.2	2.4	12.4
15	1003.0	12.5	20.0	46	S	11	WSW	46	10.4	1.2	7.4
16	1009.8	11.4	23.7	43	SE	11	S	41	12.9	5.8	0.4
17	1011.1	12.1	27.0	31	SSE	07	N	26	13.5	3.2	0
18	1002.0	16.8	24.6	36	WSW	26	WSW	67	10.0	6.8	0.2
19	1003.7	12.1	25.0	38	NW	19	W	67	7.6	7.0	0
20	1000.1	11.3	20.0	40	WSW	26	W	72	11.1	4.4	0.8
21	1010.1	10.4	22.7	37	W	11	WSW	43	9.1	2.8	1.2
22	1013.6	14.2	22.6	40	SSE	07	WSW	35	8.7	5.4	0
23	1011.8	11.4	29.2	32	SSE	06	S	43	13.4	5.6	0
24	1009.8	13.7	27.8	58	S	06	SSW	52	3.7	5.4	0
25	994.2	16.5	26.5	56	WSW	19	WSW	59	2.0	5.0	1.8
26	1012.0	12.8	18.9	66	SW	19	S	52	4.3	3.0	3.2
27	1017.6	12.2	23.2	41	ESE	13	E	41	14.1	6.6	2.0
28	1017.5	12.0	24.2	40	SE	11	ESE	48	13.4	5.4	0
29	1014.4	12.6	28.1	46	E	11	E	35	12.8	4.6	0
30	1009.6	16.0	30.3	37	S	15	S	41	13.1	5.6	0
31	1007.1	16.4	25.6	46	E	04	S	52	9.2	7.2	0
MEAN	1009.5	12.6	23.4	47					9.2	142.0	46.0
LONG TERM AVER.	1013.2	12.7	24.1	49					8.2	195.3	57.2

Appendix E
References

Clark, K. L., & F. G. McCabe, *micro-PROLOG: Programming in Logic,* Prentice-Hall, Englewood Cliffs, New Jersey, 1984

Buchanan, B. & E. Fiegenbaum, "Dendral and Meta-Dedral: Their Applications Dimension" in *Readings in Artificial Intelligence,* B. L. Webber & N. J. Nilsson (Eds.), Tioga Publishing, Palo Alto, California, 1981

Buchanan, B. & E. Shortliffe, *Rule-Based Expert Systems,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.

Duda, R., J. Gaschni & P. Hart, "Model Design in the PROSPECTOR Consultant System for Mineral Exploration" in *Readings in Artificial Intelligence,* B. L. Webber & N. J. Nilsson (Eds.), Tioga Publishing, Palo Alto, California, 1981

Fiegenbaum, E. A. & P. McCorduck, *The Fifth Generation,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1983

Hartnell, T., *Exploring Artificial Intelligence on your Microcomputer,* Interface Publications, London, UK, 1984

McCorduck, P. *Machines Who Think,* W H Freeman & Co., San Francisco, 1979

Raphael, B., *The Thinking Computer,* W H Freeman & Company, San Francisco, 1976

Rich, E. *Artificial Intelligence,* McGraw-Hill, New York, 1983

Shortliffe, E., "Consultation Systems for Physicians" in *Readings in Artificial Intelligence,* B. L. Webber & N. J. Nilsson (Eds.), Tioga Publishing, Palo Alto, California, 1981

Webber, B. L. & N. J. Nilsson, *Readings in Artificial Intelligence,* Tioga Publishing, palo Alto, California, 1981

White, D. & Shaw, W. P., *A Modern Introduction to Chemistry,* Pergamon Press, Elsmford, New York, 1980

Zadeh, L. A., "A Theory of Approximate Reasoning", *Machine Intelligence 9,* Hayes J. E., D. Michie & L. I. Mikulich (Eds.), Halsted Press, John Wiley & Sons, New York, 1979

Appendix F
Further Reading

Expert Systems and Artificial Intelligence:

Boden, M., *Artificial Intelligence and Natural Man,* Harvester Press, Basic Books Inc., New York, 1981

James, M., *Artificial Intelligence in BASIC,* Newnes Technical Books, Butterworth & Company, London, UK, 1984

Naylor, C. M. *Build your own Expert System,* Sigma Technical Press, Cheshire, UK, 1983

Pearl, J., *Heuristics — Intelligent Search Strategies for Computer Problem Solving,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984

Torrance, S. (Ed.), *The Mind and the Machine,* Halsted Press, John Wiley & Sons, New York, 1984

Simons, G. L., *Towards Fifth-Generation Computers,* NCC Publications, Manchester, UK, 1983

Simons, G. L., *Introducing Artificial Intelligence,* NCC Publications, Manchester, UK, 1984

Winston, P. H., *Artificial Intelligence,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984

Wos, L., R. Overbeek, E. Lusk & J. Boyle, *Automated Reasoning: Introduction and Applications,* Prentice-Hall, Englewood Cliffs, New Jersey, 1984

Creating Compilers:

Nicholls, J. E. *The Structure and Design of Programming Languages,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1975

Rohl, J. S. *An Introduction to Compiler Writing,* American Elsevier, New York, 1975

LISP:

Siklossy, L., *Let's Talk LISP,* Prentice-Hall, Englewood Cliffs, New Jersey, 1976

Editors (Gnosis), *Learning LISP,* Prentice-Hall, Englewood Cliffs, New Jersey, 1984

PROLOG:

McCabe, F. G., K. L. Clark & B. D. Steel, *micro-PROLOG 3.1 Programmer's Reference Manual,* Logic Programming Associates, London, UK, 1984

Pountain, D., "Prolog on Microcomputers", in *Byte Magazine,* McGraw-Hill, December, 1984

The Interface Publications Artificial Intelligence Library:

Exploring Artificial Intelligence on your Microcomputer — 1984

Exploring Artificial Intelligence on your Spectrum+ and Spectrum — 1984

Exploring Artificial Intelligence on your BBC Micro — 1985

Exploring Artificial Intelligence on your Commodore 64 — 1985 (published in the US by Bantam Books, New York, 1985)

Exploring Artificial Intelligence on your QL — 1985

Exploring Expert Systems on your Microcomputer — 1985

All titles by Tim Hartnell

Acknowledgements

Special thanks to Richard Forsyth, of Warm Boot Ltd., London, for the list of features of an expert system given in chapter one. Richard organised a seminar on artificial intelligence and machine learning which I attended in 1984, and at this seminar took notes which formed the basis of the appendix on databases, and other items of information in various places in the text. Also at this seminar, Phil Cox spoke on the Micro-Expert system which he helped develop. From Phil I gained some of the information on *logical sufficiency* and *logical necessity* which appears in chapter three.

The assistance of Melbourne businessman and statistician Nicholas McClaren, and of computer programmer and author Ross Symons, with the section on Bayes' Theorem is gratefully acknowledged.

Micro-PROLOG is marketed by Logic Programming Associates, 10 Burntwood Close, London SW18 3JU (01-874 0350). Versions are available for IBM PC; Macintosh; CP/M; plus a variety of popular home computers.

WARM BOOT LTD., is at 40 Bowling Green Lane, London, EC1R ONE (01-278 0333). Managing director is Richard Forsyth, editor of the highly recommended book Expert Systems, Principles and Case Studies **(Chapman and Hall, 1984, 0 412 26280 0)**

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

Now you can explore the fascinating world of Expert Systems on your own computer.

In this book, by one of the world's best-selling computer authors, you'll learn about the most successful Expert Systems programs developed to date, and see how the field has grown over the past decade. You can get your car started in the morning with the AUTO MECHANIC program in this book, and give yourself a quick 'stress check' with MEDICI.

The major Expert System in the book is called FUZZY RITA, which uses fuzzy logic within the framework of an Expert System shell. You can adapt it to become a genuinely-useful expert on just about any subject you choose.

You'll also have the chance to explore the languages which are beginning to dominate the Artificial Intelligence and Expert Systems' worlds. This book contains BASIC emulators of the two 'hottest' AI languages — PROLOG and LISP. There's now no need to buy these languages in order to learn them. Just type in the programs, and you'll have small-scale versions of PROLOG and LISP running so you can experiment with their power.

Machine-specific listings are provided for: SPECTRUM+ and SPECTRUM, AMSTRAD, BBC MICRO, COMMODORE 64, all MSX machines, and any computer furnished with Microsoft BASIC.

another great book from Interface

£7.95

ISBN 0-907563-74-0

9 780907 563747

EXPLORING EXPERT SYSTEMS ON YOUR MICROCOMPUTER

Tim Hartnell

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/